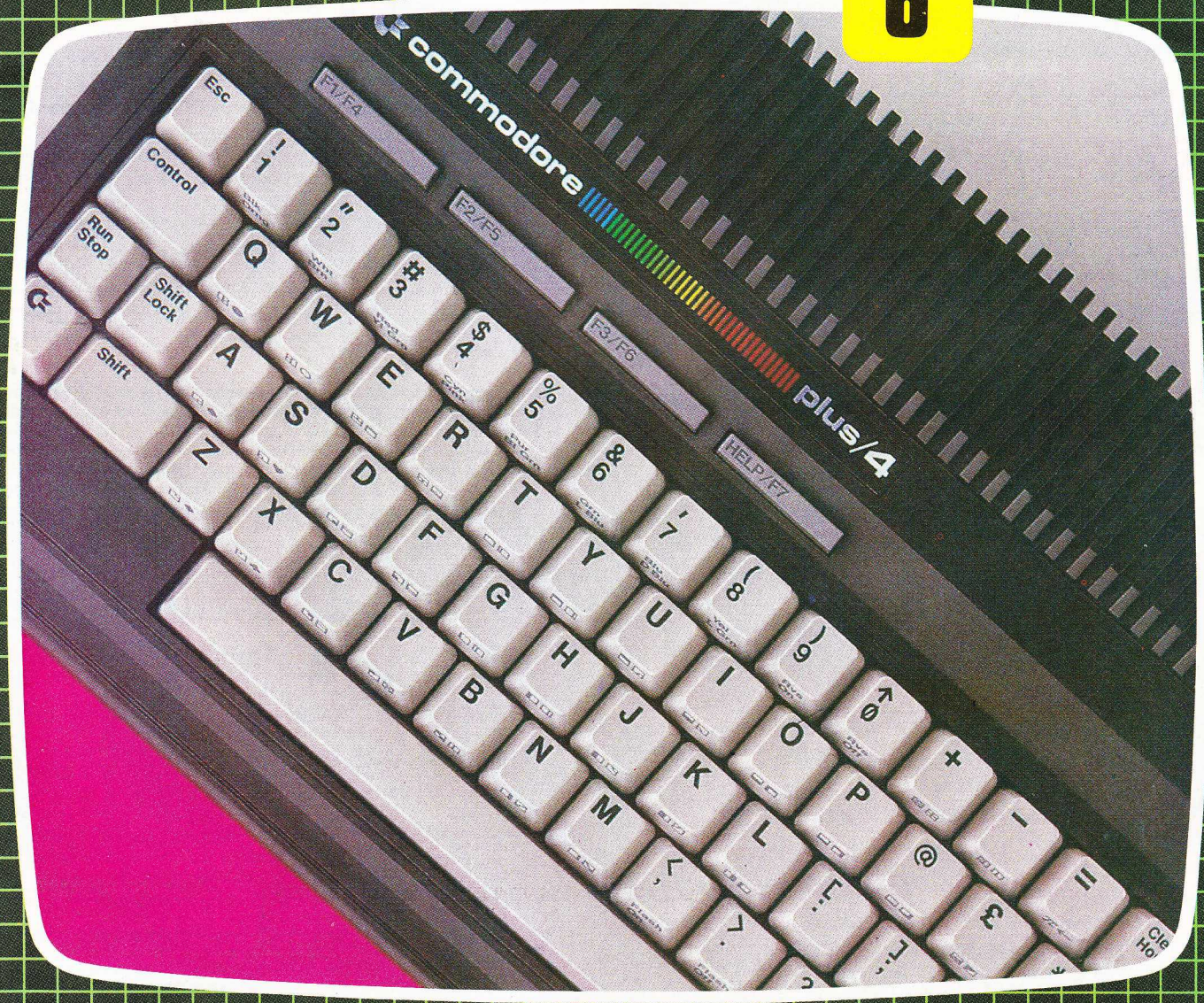
MICROCOMPUTADOR CURSO PRÁTICO

TIRE O MÁXIMO DO MICRO EM 30 SEMANAS

PUBLICAÇÃO SEMANAL ILUSTRADA

Cr$ 6.500

6



Plus/4 da Commodore • Troca de chip
Programas modulares

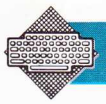
rioGráfica

MANAUS, ALTAMIRA, MACAPÁ, JI.PARANÁ, PORTO VELHO, RIO BRANCO, SANTARÉM, BOA VISTA, SINOP, ALTA FLORESTA: Cr$ 8.450

MICROCOMPUTADOR CURSO PRÁTICO

APLICAÇÕES

Fatos nas pontas dos dedos 117



PERIFÉRICOS

Tudo sob controle 120

SOFTWARE

Posições-chaves 122

PROJETOS DE PROGRAMAS

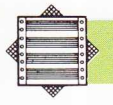
Pedalando 124

RAIO X

Plus/4 126

OFICINA

A troca do chip 128



TÉCNICAS DE PROGRAMAÇÃO

Peças de quebra-cabeça 130

LINGUAGENS

Poesia aleatória 132

PERFIL

Programas compatíveis 135

TESTE

Depois de ler com atenção todo o fascículo, responda a estas perguntas. Veja a solução no próximo número, junto com um novo teste.

1) Para acessar determinado dado num arquivo aleatório, o usuário dispõe de dois métodos. Quais são?
2) Preencha os brancos:
Comparando um banco de dados a um fichário, cada ficha seria um , e dentro dela o espaço para cada informação (nome, endereço etc.) corresponderia a um
3) Cite três dos caracteres de controle que o micro envia à impressora.
4) Qual a diferença entre as sub-rotinas do BASIC e aquelas usadas em linguagens estruturadas, como a PASCAL?
5) Elabore um procedimento em LOGO que mostre uma lista de palavras em ordem inversa. A primeira linha seria:
AP INVERTER :LISTAP

RESPOSTAS DO TESTE ANTERIOR

1) Programa heurístico é aquele que "aprende", ou seja, vai melhorando seu desempenho ao longo das tarefas que executa.
2) Os jogos de maior sucesso da Atari foram o Space Invaders e o PacMan.
3) Os computadores portáteis dispõem de processadores de texto residentes, armazenados em chips de ROM.
4) Os arquivos que permitem acesso direto a qualquer dado são os aleatórios.
5) O micro Apricot, de fabricação inglesa, possui um pequeno LCD (visor de cristal líquido) incorporado ao teclado.

PLANO DA OBRA

MICROCOMPUTADOR - CURSO PRÁTICO é uma coleção de 30 fascículos de periodicidade semanal. Os 29 primeiros terão, cada um, vinte páginas internas (miolo) e quatro capas, além de uma introdução de dezesseis páginas acompanhando o fascículo 1. O fascículo 30 terá doze páginas e quatro capas, e incluirá o índice geral da obra. Os miolos, encadernados, constituem dois volumes de um curso prático de microcomputação, com seções indicadas por tarjas de diferentes cores.
As duas primeiras capas são descartáveis; as 3.as e 4.as trazem programas de jogos. Encadernadas, formam um volume de cem páginas.

COMO ENCADERNAR

As capas duras, incluindo as guardas, estarão à venda simultaneamente com os fascículos 10 (**Volume 1**) e 25 (**Volume 2** e **Jogos**).
Volume 1 - Deve ser encadernado com os elementos dispostos nesta ordem: guardas; frontispício e introdução (publicados com o fascículo 1); miolos dos fascículos de 1 a 15; e guardas.
Volume 2 - Guardas; frontispício (que virá com o fascículo 30); miolos dos fascículos de 16 a 30; e guardas.
Jogos - Guardas; frontispício e sumário (que constituem as 3.as e 4.as capas do fascículo 30); 3.as e 4.as capas dos fascículos de 1 a 29; e guardas.

Nota: Muitos dos equipamentos que aparecem na seção RAIO X não são normalmente comercializados no Brasil. Sua divulgação é feita a título de informação ao leitor.

rioGráfica **Diretoria Executiva:** Oscar Neves; Filipe Zander; Nilo Sérgio de Almeida; Danilo Esteves Costa; **GRUPO EDUCAÇÃO E CULTURA - Planejamento e Comercialização - Diretor:** Roberto Combochi; **Gerente de Marketing:** Jaime Rodrigues; **Gerente de Produto:** José Mauro Porto; **Gerente do P.C.P.:** Aylton Menezes; **Gerente de Vendas:** Rubens Barbosa; **Coordenadores de Promoção:** Paulo Cesar M. Seixas; Edgar Mello M. Neto; **Gerente de Circulação:** Norberto Martin; **Supervisor de Operações:** Abel Pereira Pinto - **Editorial - Diretor:** João J. Noro; **Editor-chefe:** Maurício Rittner; **Administração Editorial:** Heitor de Souza Paixão; **Editor:** Jesse Navarro; **Revisores:** Isis Augusta Loyolla; Cacilda Guerra; Luiz Vicente Vieira Filho; Marcos Domingos Agathão; Maria da Graça Mendonça Couto; **Diretor de Arte:** Aníbal Monteiro; **Chefe de Arte:** Bonifácio Duardes Miranda; **Chefe de Estúdio:** José Yuji Kuribayashi; **Assistente de Arte:** Miguel Luis Escámez Simón - **Colaboradores - Editores-assistentes:** Isa Mara Lando; Luiz Carlos Pizarro Marin; Claudio Marcondes; **Texto:** Paulo Brito; **Preparação de Texto:** Luiz Carlos Cardoso; **Tradução:** Cláudio Marcondes; Maria Clara Cescato; Regina Amarante; **Pesquisa Iconográfica:** Stella Maria Quentel; **Arte:** Toshio Morobosi; **Assessoria Técnica:** Derval Junqueira de Aquino; Lauro de Lauro; Luiz Henrique Alayon; Luiz Antônio Meloni (LOGO).
As fotos não creditadas pertencem à obra original. Seguem-se os créditos das demais: Fernando Scavone. © APSIF Copenhagen, 1984; © Orbis Publishing Ltd., 1984; © Editora Rio Gráfica Ltda., 1985, para a língua portuguesa.
Composição: ERG Ltda. e AM Produções Gráficas Ltda. **Impressão:** JBIG.

NÚMEROS ATRASADOS - Você poderá comprar os exemplares que faltam em sua coleção, pelo preço do último fascículo posto à venda, em sua banca de jornal preferida. Caso não consiga obtê-los, faça sua solicitação por carta endereçada diretamente à Editora Rio Gráfica Ltda., rua Itapiru, 1209, Rio Comprido, Rio de Janeiro, CEP 20251. O atendimento será feito por via postal. Nas cidades do Rio de Janeiro e de São Paulo, as compras poderão ser efetuadas pessoalmente nos seguintes endereços: Rio de Janeiro - Rua Itapiru, 1209, Rio Comprido; São Paulo - Rua Frei Caneca, 1152, Consolação.

Distribuidor exclusivo para todo o Brasil: Fernando Chinaglia - Rua Teodoro da Silva, 907, Rio de Janeiro. Distribuidor para Portugal: Electroliber Ltda. - Rua Prof. Reinaldo dos Santos, 1488, Lisboa.

FATOS NAS PONTAS DOS DEDOS

O gerenciador de banco de dados é um dos tipos mais importantes de software para armazenamento de dados. Aqui, você vai saber em detalhes seu uso nos microcomputadores, suas limitações e como tirar vantagem dele.

Pode-se definir um banco de dados como uma coleção de informações sobre determinado assunto. Um banco de dados guarda conjuntos de informações como estas: nomes e endereços dos sócios de um clube; despesas decorrentes das reuniões do clube; datas e locais dessas reuniões; ou o pagamento das taxas de manutenção pelos sócios. Contudo, um sistema de banco de dados poderia incluir facilmente todos esses conjuntos de informações num grande arquivo.

Um arquivo é uma coleção de registros, que por sua vez abrangem diversos campos. Você pode imaginar um registro como uma ficha contendo informações sobre uma pessoa, em que os espaços para nome e endereços, por exemplo, são campos. Em alguns programas aplicativos, o software fixa a estrutura do arquivo, mas num sistema de banco de dados é mais comum escolher-se a estrutura de acordo com as tarefas. Isso implica a determinação do tamanho de cada campo e a definição do tipo de conteúdo a ser armazenado nele. Num arquivo de nomes e endereços, por exemplo, as informações sobre cada pessoa ocupam um registro: o nome é armazenado num campo de caracteres; e cada linha do endereço, em campos separados. Esses campos têm geralmente trinta caracteres.

Além dos campos de caracteres, existem também campos numéricos; o programa faz cálculos com os dados neles armazenados e dessa maneira produz informações úteis. No hipotético arquivo de sócios do clube, o programa poderia calcular quantos sócios haviam pago as taxas, qual a receita anual acumulada, ou quanto falta ser recebido. Contudo, nem todos os dados numéricos devem ser armazenados num campo numérico. Números de telefones são um bom exemplo de dados com os quais jamais se fará qualquer cálculo.

Antes de saber se um arquivo pode ser usado com eficiência em seu computador, você precisa calcular o tamanho máximo (número de registros) que ele terá e a quantidade de memória necessária para cada registro, e então determinar se o equipamento tem memória suficiente para armazená-lo.

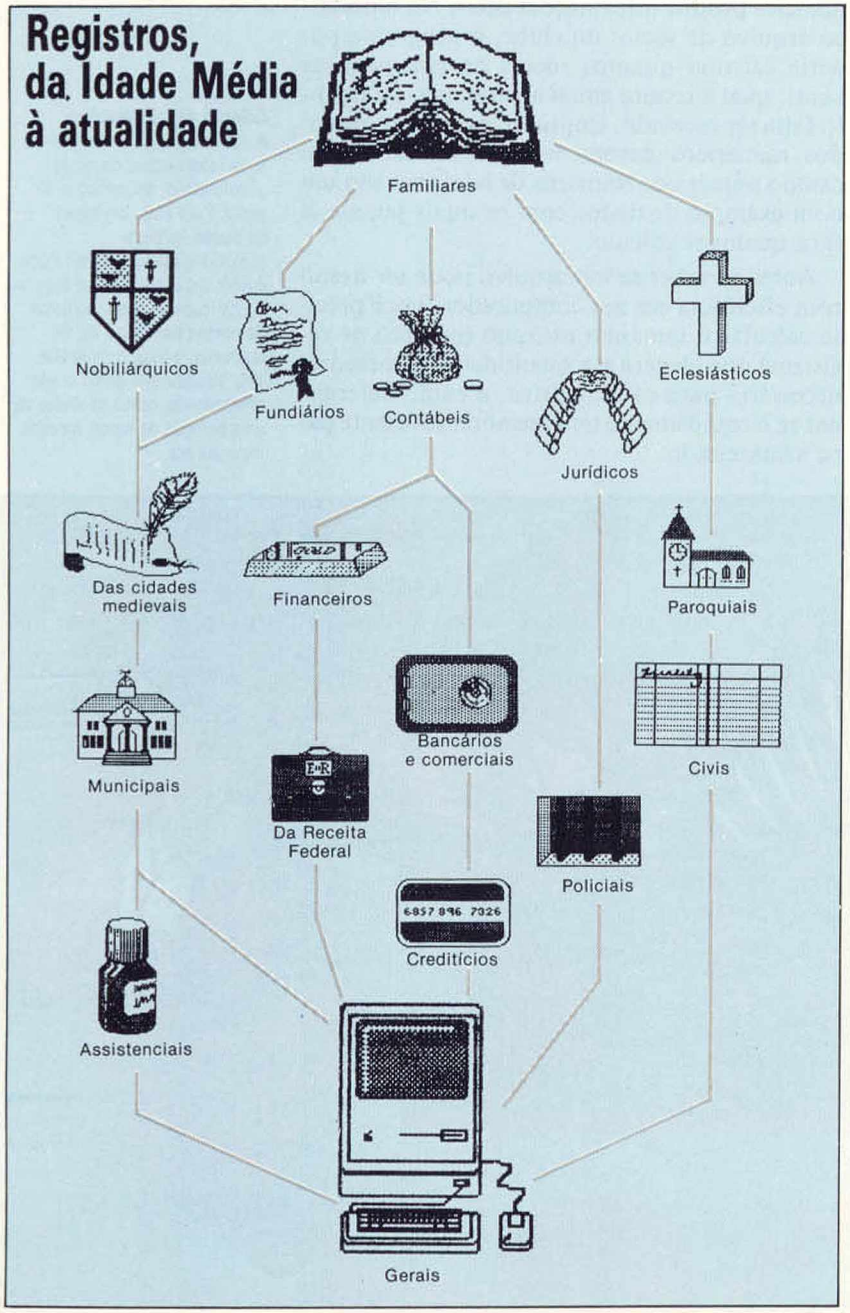
Coleção de informações
A maioria das pessoas guarda suas informações de modo desordenado, em pedaços de papel. Para elas, um banco de dados feito em microcomputador é ideal. Pode conter registros sobre o seguro do carro ou da casa, números de contas bancárias ou de telefones, e tudo o mais que seja classificável numa ordem determinada, como os dados de uma coleção de selos, moedas, minerais etc.

Ligações perigosas
No passado, os bancos de dados ficavam distanciados entre si tanto pelo espaço quanto pelo tempo. Hoje, com a possibilidade da troca de informações, chegam a ameaçar a privacidade das pessoas: bancos de dados são cada vez mais numerosos nas sociedades tecnológicas e o ritmo de acesso a eles se intensifica, porque as sociedades dependem mais e mais de informações. No quadro abaixo, algumas fases da evolução dos registros de dados.

Um registro com um nome e três campos de endereço de trinta caracteres cada, mais um número de telefone de dez caracteres, ocupa um total de 130 bytes. Se você tem um sistema de disco que armazena, digamos, 200 bytes por disco, então pode guardar até 1.500 registros em cada disquete. Num sistema com 48 bytes de memória, mas que use fita magnética, provavelmente só será possível manter cerca de trezentos registros (ocupando-se 10 bytes com o programa e o sistema operacional).

Na prática, se você deseja colocar esse arquivo em ordem ou fazer outras manipulações, precisará de memória extra; assim, convém limitar o tamanho do arquivo em fita até a metade da área disponível para ele.

Os dois sistemas de armazenamento — em disco e em fita — diferem tanto porque os arquivos em fita devem ser inteiramente passados para a memória do micro e processados em conjunto, ao passo que a velocidade e o modo de acesso aos discos permitem que os arquivos gravados neles sejam processados por etapas.



Um sistema de gerenciamento de banco de dados possibilita tirar informações de um arquivo — por exemplo, o total das despesas do ano. Ligue-o com uma informação de outro arquivo, como a receita total das taxas pagas pelos sócios, ou faça comparações entre as duas. Você pode, então, com base nessas informações, decidir o que deve ser feito: abrir a venda de títulos novos, diminuir o número de reuniões ou gastar os lucros na compra de um computador para o clube.

Ou pode querer enviar uma carta a todos os sócios, informando-os sobre a situação. Nesse caso, o banco de dados forneceria os nomes e endereços e os imprimiria diretamente nos envelopes ou em etiquetas adesivas. Ou, então, ele poderia ser ligado a um programa de processamento de texto, para escrever cartas ou informes, fazendo automaticamente o endereçamento.

Embora muitas coisas possam ser feitas por um gerenciador de banco de dados ideal, não é fácil encontrar um que venha com todas as facilidades juntas, sobretudo se você tiver um computador de uso doméstico com espaço restrito de memória e armazenamento em fita ao invés de em disco.

Muitos dos pacotes de bancos de dados menos sofisticados trabalham com apenas um campo por vez, de maneira que, mesmo podendo guardar diferentes conjuntos de informações, você não poderá ligá-los ou compará-los no verdadeiro estilo do banco de dados.

Um perigo real em todo processamento de dados é colocar sua preciosa informação num computador e então, por acidente, apagá-la. Ao lidar com informações importantes, é fundamental manter cópias de segurança e imprimir os dados sempre que houver atualizações, para que nada possa ficar irremediavelmente perdido.

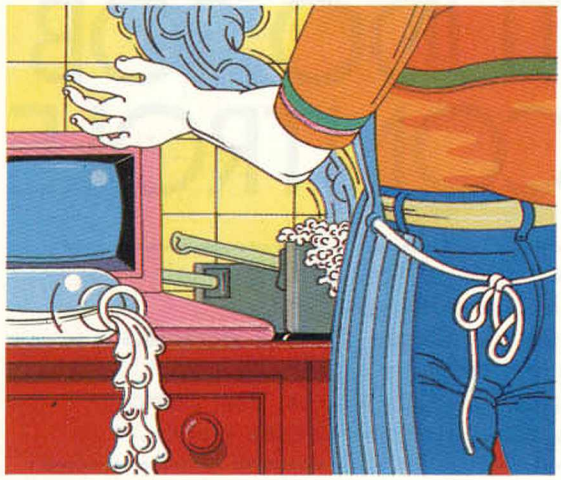
Um pacote simples de banco de dados processa um arquivo por vez. O sistema deve fazer o trabalho de rotina de colocar a informação no arquivo tão facilmente quanto se preenche um formulário, e deve permitir que você tenha acesso à informação tanto na tela quanto por meio da impressora. O usuário seleciona registros por meio de critérios de busca e seleção, como, por exemplo, "todos os sócios que não pagaram este ano e têm o CEP xis".

Um pacote deste tipo também deve permitir a impressão de campos como os de nomes e endereços, para a emissão de etiquetas de endereçamento. Esse tipo mais simples de pacote existe para a maioria dos computadores de uso doméstico que funcionam com fita. No entanto, se você tem um sistema de disco, a variedade de software disponível é muito maior.

Pacotes mais sofisticados permitem que você desenhe um formulário de entrada de dados na tela para combinar com um sistema anterior ba-

Receita que desanda

Heitor Megabyte tem um pequeno micro, gosta muito dele e de mostrar tudo o que sabe. A máquina ajudou-o a entender os computadores e a aprender um pouco de programação em linguagem BASIC, mas, infelizmente, Heitor ainda não sabe a diferença que existe entre aplicação séria e diversão. Ele insiste em usar o computador para guardar os números dos telefones de seus (poucos) amigos e tem um software para banco de dados em fita de que se vale para guardar receitas culinárias. Quando Heitor convida alguém para jantar, a comida, invariavelmente, só sai por volta da meia-noite, porque ele insiste em conferir os detalhes de cada receita do micro. Para isso, tem de ligar o computador na tomada com gestos teatrais, carregar o programa em fita (o que em geral exige várias tentativas), esperar durante intermináveis minutos enquanto o micro procura a receita, ler as instruções na tela, aos poucos, e finalmente preparar o jantar. Não adianta chegar mais tarde, porque Heitor não começa a preparar a comida sem a exibição. Para quem quer ver um micro gerando eficiência, é uma lástima.



Escolha sua música

Faz tempo que a coleção de discos de Clara Chip é grande. E ela resolveu aproveitar esse fato para alugá-los. A experiência ensinou-lhe que é fundamental oferecer os discos certos a cada tipo de ouvinte, e por isso ela decidiu comprar um computador. Para ter velocidade e alta capacidade de armazenamento, escolheu um sistema com dois drives. Então, escreveu num papel os nomes das categorias de música que gostaria de incluir em seu software de banco de dados. Havia dois grupos principais (música erudita e música popular), cada um com subtítulos, que eram também subdivididos. Clara usa o gerenciador de banco de dados para escolher os discos de acordo com o cliente — digitando, por exemplo, o comando para listar todos os discos de samba e então fazendo uma seleção mais detalhada, em nível de bossa nova ou carnaval. Ela examina a listagem no monitor e em seguida tira uma cópia dela na impressora. Depois, é só ir à estante e apanhar o disco escolhido para o cliente.

seado em papel. Outros pacotes possibilitam selecionar itens numéricos (por exemplo: maiores ou menores que determinada quantia). Portanto, é importante procurar um pacote adequado a suas necessidades específicas.

Arquivos interligados

Um computador com gerenciador de banco de dados é útil para tarefas que envolvem repetição, ou nos casos em que uma grande quantidade de informação precisa ser armazenada ou examinada rapidamente. Mas há algumas aplicações que não são adequadas para os sistemas de banco de dados: não seria sensato vasculhar a lista telefônica de um computador doméstico toda vez que você quisesse fazer uma ligação. No tempo gasto para ligar o computador, carregar o programa de banco de dados, chamar o arquivo e procurar o número, você poderia ter feito seis ligações, pegando os números de uma agenda.

Um pacote de banco de dados mais potente proporciona todas essas facilidades e você ainda pode usar a matemática para obter dados tais como a renda total proveniente das taxas de manutenção do clube, ou calcular qual a instalação mais usada nele.

Vários arquivos de informações podem ser interligados, de modo que os dados sobre assuntos correlatos sejam usados em conjunto. Por exemplo, determinado jogador pertencente a nosso clube imaginário tem seus dados pessoais registrados num arquivo e os detalhes de seu desempenho nos torneios do clube em outro. Quando for pedido um histórico sobre ele, os dois arquivos poderão ser usados em conjunto. Para se ter uma posição do desempenho de todos os jogadores do clube em determinada data, consulta-se o arquivo chamando-se todos os registros daquela data em vez de todos os registros de cada pessoa.

Para executar um programa com facilidades tão extensas, seria necessário usar um computador de mesa ou profissional, com um mínimo de 64 Kbytes de memória e sistema de disco.

Os gerenciadores de bancos de dados têm condição de processar qualquer tipo de informação, desde simples listas até sistemas altamente complexos, com arquivos múltiplos, e têm facilidade para criar relatórios elaborados e capacidade de processar informação de um modo que pode ser transferido para outro software, com os processadores de textos.

TUDO SOB CONTROLE

Até a impressora matricial mais simples pode gerar uma série de efeitos gráficos especiais que melhoram a apresentação visual de documentos. Veja como conseguir alguns desses efeitos.

Impressoras do tipo matricial fazem muito mais do que produzir listagens de programas. Uma variedade de efeitos especiais também pode ser executada por elas.

Até a impressora mais simples permite que se altere o tamanho dos caracteres impressos. Em geral, um texto é impresso com oitenta caracteres por linha, número que pode ser aumentado, selecionando-se o modo "impressão condensada" (que usa caracteres menores), ou diminuído, escolhendo-se a "impressão ampliada". O espaço entre as linhas se altera da mesma forma.

A impressora que examinaremos em detalhes é um bom exemplo de máquina com grande variedade de tipos de impressão. Você pode optar pela dupla impressão da letra (negrito), que imprime o texto com caracteres mais escuros, ou pela impressão em itálico. Uma de suas características mais interessantes é a capacidade de mudar qualquer dos caracteres armazenados na sua memória, o que facilita, por exemplo, a impressão de símbolos científicos. Para começar, vamos ver como uma impressora realiza a simples tarefa de imprimir a listagem de um programa.

A maneira pela qual um computador "fala" com uma impressora varia conforme a máquina. O Dragon, por exemplo, usa uma variação simples do comando LIST — o LLIST —, instruindo a impressora para produzir uma cópia do programa. Outras máquinas precisam da abertura de "canais" ou "correntes" para ter acesso à impressora. Para saber como realizar essa operação, o mais indicado é consultar o manual do computador.

Estabelecida a comunicação entre as duas máquinas, prepare-se para enfrentar alguns problemas. Os mais freqüentes são o do texto ilegível ou o das linhas em branco entre as linhas do programa. A explicação para esses defeitos está na diferença entre um caractere "avanço de linha" (Line Feed ou LF) e um caractere "retorno do carro" (Carriage Return ou CR). Depois que envia uma linha do texto para a impressora, o computador também envia um caractere de retorno do carro que faz a cabeça de impressão voltar para a margem esquerda, pronta para imprimir nova linha. Alguns computadores também enviam um caractere de avanço de linha para mover o papel para a linha seguinte; outros, não, presumindo que a impressora faça isso automaticamente. A maioria das impressoras possui um interruptor interno que define se a impressora gera seus próprios sinais de avanço de linha. Se ocorrerem alguns desses problemas, mude o interruptor para a posição alternativa.

Além de produzir listagens de programas, uma impressora pode ser usada como dispositivo de saída — em vez de apresentados na tela, os caracteres são impressos no papel. A operação desse processo também varia de um computador para outro. Os exemplos de programação aqui apresentados usam LPRINT, e talvez você tenha de fazer uma alteração para adaptá-los a sua máquina.

Etiquetas de endereçamento

```
10  LPRINT "SR. L. HENRIQUE ALAYON"
20  LPRINT "AV. FARIA LIMA, 1106/709"
30  LPRINT "01452  SAO PAULO     SP"
40  FOR I = 1 TO 8
50  LPRINT
60  NEXT I
70  GOTO 10
```

Esta listagem é um programa simples para produzir etiquetas de endereçamento. Como não usa código especial de controle, o programa funcionará com qualquer marca de impressora. Como está, o programa imprimirá o mesmo nome e endereço repetidamente.

Talvez você queira alterar isso, para colocar nomes e endereços diferentes, ou mesmo para que eles sejam lidos de um arquivo de dados. O loop FOR-NEXT entre as linhas 50 e 70 imprime sete linhas em branco e é usado para posicionar a cabeça da impressora corretamente no começo de cada etiqueta. O número exato de linhas em branco precisa ser ajustado para sua máquina.

Nosso programa é adequado para etiquetas de impressão simples, mas, para imprimir algo mais complexo, como uma fatura ou um cabeçalho, será preciso usar alguns efeitos especiais. Eles são produzidos enviando-se códigos de controle para a impressora e também para os caracteres normais do texto.

Além de um código para cada caractere no teclado, o Padrão ASCII tem um grupo de caracteres "invisíveis", que não imprimem nada na tela ou no papel. São estes os códigos usados para produzir os efeitos especiais da impressora: no conjunto ASCII padrão há quatro códigos

ELITE

```
ABCDEFGHIJKLMNOP
QRSTUVWXYZabcdef
ghijklmnopqrstuv
wxyz0123456789 !
"£$%&'()*+,-./:;
<=>?@[\]^_`{|}~
```

PICA

```
ABCDEFGHIJKLM
NOPQRSTUVWXYZ
abcdefghijklm
nopqrstuvwxyz
0123456789 !"
£$%&'()*+,-./
:;<=>?@[\]^_`
{|}~
```

EMPHASISED PICA ITALIC

```
ABCDEFGHIJKLM
NOPQRSTUVWXYZ
abcdefghijklm
nopqrstuvwxyz
0123456789 !"
#$%&'()*+,-./
:;<=>?@[£]^_`
{|}~
```

ENLARGED ELITE

```
A B C D E F G H
I J K L M N O P
Q R S T U V W X
Y Z a b c d e f
g h i j k l m n
o p q r s t u v
w x y z 0 1 2 3
4 5 6 7 8 9  !
" £ $ % & ' ( )
* + , - . / : ;
< = > ? @ [ \ ]
^ _ ` { | } ~
```

DOUBLE-STRIKE CONDENSED PICA

```
ABCDEFGHIJKLMNOPQRSTUVW
XYZabcdefghijklmnopqrst
uvwxyz0123456789 !"£$%&
'()*+,-./:;<=>?@[\]^_`{
|}~
```

Pontilhados
A impressora matricial possui uma variedade de tipologias (Pica, Elite e Italic) e modelos de tipos (condensados, ampliados e com dupla impressão). Os exemplos acima foram impressos na Epson FX-80.

(17, 18, 19 e 20) que são reservados para comandos de controle de dispositivos. O conjunto de caracteres ASCII não possui reserva suficiente de comandos de controle para as setenta e tantas funções de uma Epson FX-80. A fim de resolver esse problema, a maioria dos efeitos é produzida enviando-se "códigos de escape" para a impressora. Eles consistem em dois ou mais códigos de caracteres, começando com um caractere ESC (código ASCII 27). Por exemplo, para ligar a função de espaçamento proporcional numa Epson, você envia ESC-p — isto é, o caractere Escape seguido do caractere "p" minúsculo.

Em BASIC, isso é escrito assim:

```
LPRINT CHR$(27);"p"
```

O caractere ESC não pode ser produzido normalmente apertando-se a tecla [Escape]. Por isso usa-se a função CHR$.

No BBC, você usaria:

```
VDU2
VDU1, 27,1, 112
VDU3
```

O comando VDU2 liga ("ativa") a impressora; VDU1 significa "envie o seguinte caractere somente para a impressora". PRINT também mandaria o caractere ESC para a tela, mas sem o mesmo resultado. VDU3 desliga ("desativa") a impressora.

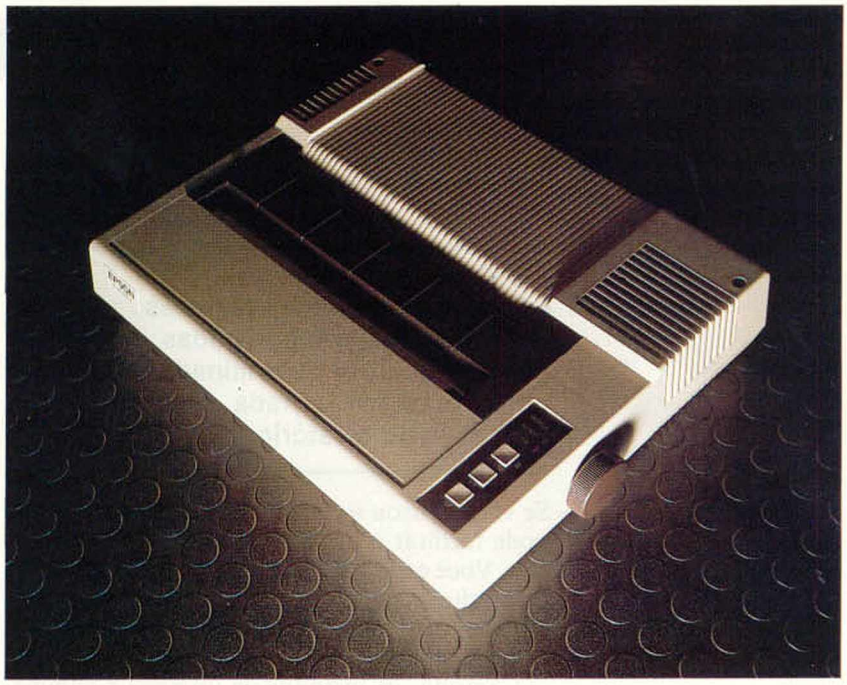
Essas seqüências de comando aplicam-se unicamente à Epson FX-80. Se você tentar a mesma seqüência de código para uma impressora diferente, ela não produzirá efeito, poderá acontecer algo inesperado ou até desligar a impressora (isto é, ela se recusará a responder ao computador).

Criando uma fatura

Nossa segunda listagem demonstra o uso de algumas das funções da Epson para criar o cabeçalho de uma fatura, usada por uma pequena oficina. Os códigos aqui aplicados são os da Epson FX-80. Se sua impressora for incompatível com a Epson, você deve adequar os códigos de controle.

```
999  REM *** CABECALHO DE FATURA ***
1000 LPRINT CHR$(12)
1010 LPRINT CHR$(14);TAB(12);"ABC
     MOTORES LTDA."
1020 LPRINT CHR$(13);CHR$(13)
1030 LPRINT CHR$(27);"E";
1040 LPRINT TAB (36);
1050 LPRINT CHR$(27);"-";CHR$(1);
1060 LPRINT "FATURA";
1070 LPRINT CHR$(27);"-";CHR$(0);
1080 LPRINT CHR$(27);"F";
1090 LPRINT CHR$(13);CHR$(13);CHR$(13)
```

Para começar, a linha 1000 envia o caractere com o código 12 para a impressora. Este é o caractere de "avanço de página" (FF), que dá instruções à impressora para correr o papel até o começo de uma nova folha. Depois, temos o código ASCII 14: ele é chamado de caractere de "shift out" (SO) e, na Epson, faz com que todo o texto subseqüente seja impresso em letras ampliadas. Em nosso programa, é usado para o cabeçalho, escrevendo o nome da oficina em letras grandes. A função TAB é usada para centralizar o cabeçalho.



Epson FX-80
É uma impressora popular entre os usuários de micros comerciais e de emprego doméstico. Tem uma cabeça de nove agulhas e sua velocidade máxima de impressão chega a 160 caracteres por segundo. No Brasil, uma impressora similar à Epson é a Grafix.

CHR$(13) é o caractere de retorno do carro (CR), que produz uma linha em branco quando impresso por si só. É usado várias vezes entre as linhas 1020 e 1090, para espaçar o cabeçalho da fatura. ESC-E, na linha 1030, liga o modo de dupla impressão: com isso, o texto que vem a seguir é impresso em tipos mais escuros. A linha 1050 liga a função de "sublinhar" e a linha 1070 desliga-a, depois de imprimir e sublinhar a palavra "FATURA". ESC-F anula o modo de dupla impressão. O texto impresso ficará assim:

```
ABC MOTORES LTDA.

FATURA
```

Essa é apenas a parte inicial. Um programa completo de fatura incluiria linhas para imprimir detalhes do cliente, como nome, marca do carro, valor a pagar etc. Esses detalhes seriam obtidos a partir de uma série de perguntas feitas no início do programa e as respostas ficariam armazenadas como variáveis.

Os dois programas aqui apresentados são exemplos simples dos usos alternativos de uma impressora do tipo matricial. Assim, programar sua impressora pode ser tão divertido quanto programar o próprio computador.

POSIÇÕES-CHAVES

A indexação de registros e o hashing são duas técnicas de acesso muito empregadas para se localizar determinada informação armazenada num registro aleatório.

Se você já usou o acesso aleatório, sabe que ele pode facilitar o uso de arquivos em programação. Você especifica qualquer registro a ser lido ou escrito, sem ter de preocupar-se com os complicados procedimentos necessários para recuperar informações armazenadas seqüencialmente.

Contudo, os métodos de inserção e eliminação não são os mais eficientes para usar arquivos. Um método bem melhor de acesso à informação é o índice.

Para criá-lo, determinado campo de cada registro deve ser especificado como *chave*. O conteúdo desse campo será usado para localizar o registro para consulta ou processamento. Assim, os valores de tais campos-chaves de cada registro, junto com o número desse mesmo registro, formam um índice. Desse modo, se o registro, digamos, de Leila Brito, tem o número 17 e é o oitavo numa listagem em ordem alfabética, então o algoritmo (a fórmula) do índice colocará o número 17 na oitava posição dessa listagem. Se o índice for periodicamente ordenado, a localização de um registro poderá ser um processo muito rápido.

Arquivos índices

Em geral, o índice fica na memória do computador durante o processamento, para que seu acesso seja imediato. Ele é gerado em instantes, por uma rotina que lê o arquivo inteiro, trazendo o conteúdo de cada campo-chave para uma matriz. Esse índice pode ser ordenado para uso imediato, mas isso implica grande demora.

Uma saída é armazenar vários índices como arquivos, no disco, junto com arquivos de dados. Dessa forma, cria-se qualquer número de campos-chaves para um registro, gerando para cada tipo de campo um arquivo índice guardado em disco. O arquivo pode ser indexado de várias maneiras: por exemplo, com os registros organizados em ordem alfabética (por nome), cronológica (por data) e assim por diante.

E como se armazena um arquivo índice? Para cada registro indexado ele precisa conter dois campos: a informação do campo-chave e o número do registro. Isso será transferido do disco para a memória, pronto para uso.

Esta é uma excelente aplicação para um arquivo seqüencial, no qual, ao contrário do que acontece no aleatório, a informação é solicitada na ordem em que foi armazenada. Nesse ponto, os arquivos de acesso seqüencial e os de acesso aleatório se complementam.

A remoção de registros inúteis nos arquivos aleatórios indexados é uma questão de marcá-los como anulados e assegurar-se de que eles não mais constem do índice. A maneira mais simples de fazer isso é colocar o sinal de "anulado" (um asterisco no começo do primeiro campo do registro, digamos). Com isso, a chave para esse registro pode ser removida do índice; ou, ainda, o número do registro, receber um valor especial, indicando anulação.

Qualquer que seja o método escolhido, porém, é importante que haja um controle dos registros anulados. Quando um novo for acrescentado ao arquivo, pode ficar sobreposto a um anulado. Nesse caso, a entrada do índice original no disco deve ser substituída por uma nova, e num ponto conveniente; e o índice deve ser reordenado, para incluir o novo registro na posição correta dentro do arquivo. Dessa forma, um programa permite a recuperação de registros anulados por acidente (desde que nada tenha sido escrito sobre eles).

Convém providenciar uma forma de agrupar os registros do arquivo de índices, porque o sistema que apresentamos permite que eles sejam armazenados fora de ordem, com numerosos e desnecessários intervalos entre um e outro. À medida que o arquivo cresce, a velocidade de acesso diminui. A rotina de agrupamento deve colocar os registros numa ordem conveniente e eliminar os anulados.

O hashing

A indexação não é a única maneira de localizar registros num arquivo com grande rapidez. Um método alternativo é o hashing, adequado para arquivos muito grandes e que, por essa razão, só é encontrado em sistemas de discos rígidos ou em máquinas com discos flexíveis de alta capacidade. Contudo, muitos sistemas operacionais e programas usam o hashing para acelerar as operações.

O hashing substitui o índice por um algoritmo. Ele toma o valor contido no campo-chave e a partir dele produz um número (o hash). O registro é então armazenado na posição do arquivo indicada por aquele número.

Planeja-se a fórmula de acordo com o tipo de informação do campo-chave. Se ele contém a data, por exemplo — permitindo que os registros

sejam organizados cronologicamente —, pode-se usar o número do mês, multiplicado pelos dois últimos dígitos do ano, mais o número do dia, para produzir o hash.

Suponha que desejamos criar um arquivo com o hash dos registros de empregados, usando sobrenomes como chave. O algoritmo que se pode usar é o seguinte: pegue os códigos ASCII das quatro primeiras letras e trate-os como um número de oito dígitos; eleve esse número ao quadrado e então use os quatro últimos dígitos do resultado como hash. JONES, portanto, é registrado como 1161, enquanto JONQUIL é registrado como 0161.

O hashing é muito diferente de um sistema de indexação. Com ele, você só pode ter um campo-chave (e um algoritmo) por arquivo; e ele é usado toda vez que se coloca um registro no arquivo.

O hashing mostra-se menos flexível do que a indexação, porém mais rápido. Para encontrar determinado registro, o programa apenas pega a chave, gera o hash e recupera o registro desejado.

Um problema ocorre quando dois registros geram o mesmo código e portanto deveriam ocupar a mesma posição num arquivo. Para evitar isso, os algoritmos são cuidadosamente planejados para que jamais duas chaves (a não ser as idênticas) gerem o mesmo hash. Além disso, os registros são espaçados no arquivo, de modo que dois hashes, aparentemente próximos um do outro, na realidade fiquem separados por uns cinco registros sem uso.

Quando se armazena um registro, sua chave é tratada para produzir um número hash de registro. Se o registro estiver ocupado, o sistema procura o registro seguinte pela ordem seqüencial. Isso pode ser feito com um bloco inteiro de uns cinco registros associados com aquele número hash.

Se um registro deve ser localizado, sua chave é tratada pelo algoritmo do hashing e aquele grupo de registros é então examinado seqüencialmente até encontrar-se a combinação perfeita. Isso parece anular a vantagem da velocidade, mas o hashing reduz o número de registros a serem examinados de aproximadamente 3 mil para apenas cinco ou seis.

O hashing também acelera a anulação de registros. Você faz o hash da chave do registro, dá uma busca rápida para localizá-lo e marca sua posição como vazia. Então, da próxima vez que um registro com idêntico hash for acrescentado ao arquivo, ele será escrito por cima do anulado.

E o que acontece quando os cinco registros para determinado hash estiverem ocupados? Há diversas maneiras de resolver isso. A mais óbvia consiste em enviar uma mensagem de arquivo cheio. Freqüentemente, os registros que não cabem são colocados num arquivo de excedentes, com seu próprio índice, e incorporados ao arquivo principal quando possível. Na maioria dos sistemas, há grande empenho em evitar excedentes, mantendo-se arquivos hash com menos de 80% de sua capacidade total.

Índice acoplado

Localize "Davi"

Chave	N.º
Aníbal	1
Beatriz	–1
Bonifácio	5
Cláudia	–1
Davi	7
Donato	23
Fábio	15
Gregório	28
Heitor	37
João	25
Klaus	11
Maíra	10

Arquivo índice

Um índice é o modo mais comum de orientar a busca de registros dentro de um arquivo aleatório: é uma lista de valores contidos em determinado campo de cada registro, armazenada na memória do computador enquanto o arquivo está sendo usado. Desse modo, pode ser examinada instantaneamente. Neste exemplo, registros inúteis ganharam o número –1, para serem depois substituídos por novos registros.

Arquivo principal

	Nome	Tel. com.	Tel. res.	Profissão
1	Aníbal	242 0791	727 0942	Desenhista
2	Pedro	636 2418	221 3940	Contador
3	Samuel	631 0836	286 8170	Editor
4		Registro cancelado		
5	Bonifácio	729 8213	236 2190	Fornecedor
6	Paulo	836 6622	298 4316	Decorador
	Davi	743 7216	450 6926	Jardineiro
8		Registro cancelado		
9		Registro cancelado		
10	Maíra	730 6321	429 7592	Estoquista
11	Klaus	493 9899	455 8431	Advogado
12	Wagner	736 7700	693 0452	Cabeleireiro

O hashing

Localize "Davi"

ALGORITMO DO HASHING

O hashing usa o conteúdo de uma chave para achar um grupo de registros, entre os quais está o procurado. Feito isso, a localização dele é instantânea.

O acréscimo de novos registros é feito nos espaços entre os blocos.

Nome	Tel. com.	Tel. res.	Profissão
Daniel	629 0491	430 0592	Funileiro
Dimas	436 2488	362 0066	Jornalista
Darci	730 0021	626 9191	Faxineira
Doroti	439 9933	630 4918	Psicóloga
Davi	743 7216	450 6926	Jardineiro
Dráusio	830 0123	340 9924	Enfermeiro
Ernesto	731 6666	458 0021	Desenhista
Eraldo	831 8294	450 6218	Dentista

Arquivos construídos com base em hashing oferecem alta velocidade de acesso aos registros, mas têm restrições que exigem muito cuidado na programação. Em troca, reduzem de 3 mil para cinco ou seis registros o volume de busca, quando se compara, digamos, com uma busca seqüencial. Para acrescentar, consultar ou eliminar registros, a velocidade de acesso ao arquivo é sempre muito elevada.

PEDALANDO

Na maioria, os jogos famosos são longos e escritos em código de máquina, devido à velocidade. Mas é possível criar um jogo em BASIC que diverte e desafia a rapidez de seus reflexos.

Baseado numa cena do filme *Tron*, produzido pelos estúdios de Walt Disney, este jogo — Pedalando — requer de você e de seu oponente muita habilidade e reações rápidas. Desenvolve-se numa arena e o único controle existente permite girar sua bicicleta num ângulo de 90° à velocidade máxima.

Os dois veículos deixam um rastro de luz e o objetivo do jogo é forçar o adversário a bater no labirinto cada vez mais estreito que você cria à medida que gira em torno da arena.

O jogo foi feito para o CP 400, cujo BASIC é lento, mas o programa está escrito em termos de velocidade e não de elegância, evitando chamadas de sub-rotinas, pois elas prejudicariam a rapidez da execução.

A primeira etapa — bastante simples — consiste no desenho da arena e do placar. O único ponto a observar é que a borda da arena fica um caractere para dentro da área útil da tela. Isso serve para assegurar que os gráficos resultantes de uma colisão com a parede da arena não saiam da tela:

```
10 P=0:Q=0:V1=0:V2=0
20 PT=-1:UM=0
100 PMODE3,1:PCLS:CLS:SCREEN0,0
110 PRINT@170,"BICICLETA 1 =";Q
120 PRINT@202,"BICICLETA 2 =";P
130 COLOR 4,1
140 LINE(1,1)-(254,190),PSET,B:
    LINE(0,0)-(255,191),PSET,B
150 REM
```

A arena foi desenhada em vermelho; o amarelo representa a bicicleta 1; e o azul, a bicicleta 2. As variáveis p e q registram a contagem do jogo para os dois competidores.

O passo seguinte consiste em inicializar todas as variáveis, e aqui é preciso pensar de que maneira vamos implementar a ação principal do jogo. A ação para uma bicicleta sozinha é simples e aparece claramente no fluxograma. Usando PPOINT, verificamos se a atual posição da bicicleta está ocupada; se estiver, passamos para a rotina de colisão. Se não estiver, passamos para aquela posição usando LINE e lemos o teclado para verificar se houve alguma mudança de direção.

A posição é então aumentada de uma unidade em nossa direção atual, e o ciclo recomeça. Portanto, precisamos de quatro variáveis: duas para as atuais coordenadas x e y e duas para nossa direção atual ao longo dos eixos x e y.

Já que se trata de duas bicicletas andando ao mesmo tempo, uma solução elegante seria usar quatro arrays de dois elementos cada, x(2) e y(2), para as posições do exemplo. Mas isso tornaria o jogo lento e, portanto, temos de usar oito variáveis separadas:

```
200 X=40: Y=88
210 M=215: N=88
220 A=1: B=0
230 I=-1: J=0
```

Isso determina as posições iniciais das bicicletas e coloca-as em movimento, uma indo em direção à outra, 1 pixel por vez. A ação básica é então simples de implementar:

```
400 IF PPOINT(X+A,Y+B)=3 OR
    PPOINT(X+A,Y+B)=4  THEN
    C0=2:GOTO700
410 IF PPOINT(M+I,N+J)=2 OR
    PPOINT(M+I,N+J)=4  THEN
    C0=3: X=M: Y=N: GOTO700
420 PSET(X,Y,2): PSET(M,N,3)
```

As linhas de 500 a 570 constituem a rotina de teclado que estabelece novos valores para A, B, I e J.

```
600 X=X+A: Y=Y+B
610 M=M+I: N=N+J
620 GOTO400
```

O único ponto que pode ser confuso é a linha 410, em que as variáveis da bicicleta 1 estão determinadas para a bicicleta 2, e uma variável nova, co, é introduzida. Isso se faz para que uma única rotina possa ser usada para a ação de colisão, onde x e y são usados simplesmente para indicar o ponto em que a colisão ocorre, e co determina a cor.

Na rotina para verificar o teclado, em vez das afirmações IF-THEN, que são lentas, podemos usar o comando rápido INKEY$ para ler as teclas escolhidas, Q e A, que controlam o movimento para cima e para baixo da bicicleta 1, e P e ENTER para a bicicleta 2. À direita e à esquerda são X e C para a bicicleta 1 e N e M para a bicicleta 2.

```
500 T$=INKEY$:IF T$="A" THEN A=0:
    B=1
510 IF T$="Q" THEN A=0: B=-1
520 IF T$="X" THEN A=-1: B=0
530 IF T$="C" THEN A=1: B=0
540 IF T$=CHR$(13) THEN I=0: J=1:
    UM=1
550 IF T$="P" THEN I=0: J=-1: UM=1
560 IF T$="M" THEN I=1: J=0: UM=1
570 IF T$="N" THEN I=-1: J=0: UM=1
```

Resta agora a rotina de colisão e atualização do placar. Escolhe-se uma série expansiva de círculos concêntricos, centralizados no ponto de impacto, com raios de 4, 6 e 8 pixels:

```
700 FOR D=1 TO 3
710 CIRCLE(X,Y),2+D*2,CO
720 NEXT D
730 IF CO=2 THEN P=P+1: GOTO750
740 Q=Q+1
750 FOR Z=1 TO 200: NEXTZ: GOTO100
```

Isso encerra o jogo, com a última afirmação voltando em loop para os procedimentos iniciais. No entanto, o jogo poderia começar com um procedimento inicial que o tornaria mais interessante:

```
300 T$="":PRINT@448,"USE QUALQUER
    TECLA PARA  INICIAR";
310 T$=INKEY$: IF T$="" THEN 310
320 T$="":SCREEN 1,0
```

Isso lhe dá um intervalo entre rodadas consecutivas. Agora, só falta gravar o programa usando SAVE (para disco) ou CSAVE (para cassete). Introduzindo-se a linha 5 RUN, o jogo será executado automaticamente assim que for carregado.

O jogo se tornaria mais excitante com telas de instruções, telas de carga, opção para um só jogador com uma rotina estratégica para controlar a outra bicicleta, som e gráficos mais complexos. Certamente, porém, essas alternativas viriam em prejuízo da rapidez da ação.

Para um só jogador

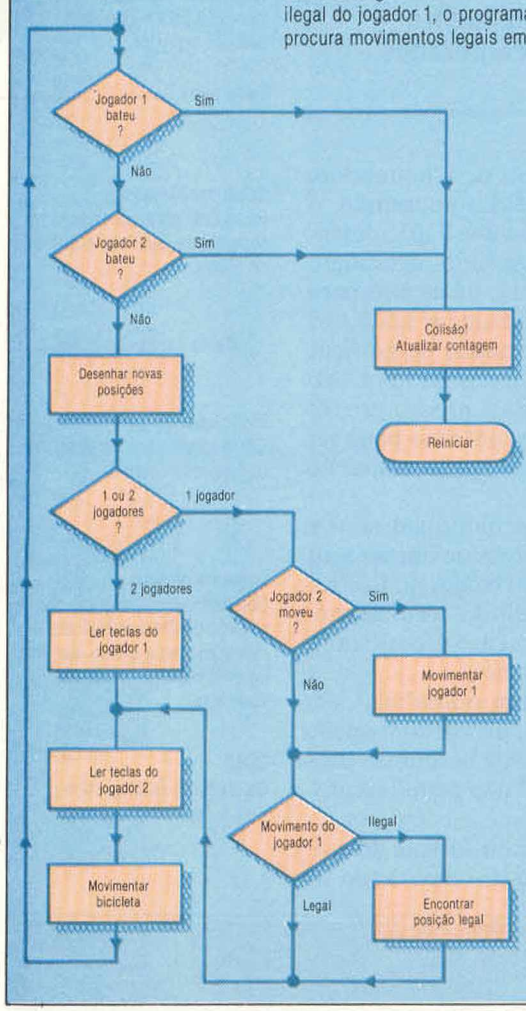
Encaixando

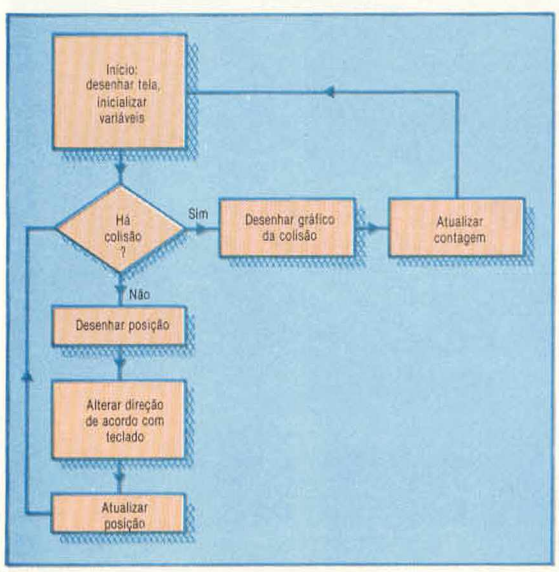
Na versão individual, o computador é o jogador 1. Ignora-se a parte do programa que verifica as teclas de comando do jogador 1, e o algoritmo no código dado permite que o jogador 1 continue a andar em linha reta até que isso gere um movimento ilegal, ou que o jogador 2 ande. Neste último caso, o movimento é duplicado e sua legalidade é verificada. Quando se gera um movimento ilegal do jogador 1, o programa procura movimentos legais em ângulos retos com a direção do movimento ilegal; se nenhum for encontrado, então a posição é crítica. Há muitas maneiras de implementar uma versão desse jogo para um só jogador. As modificações que sugerimos possibilitam escolher entre as versões para um e dois jogadores no começo de cada partida. Não é difícil obter algoritmos satisfatórios para este jogo, mas há dificuldades para implementá-los em BASIC sem diminuir consideravelmente a velocidade. No programa, eis as modificações:

```
260 PRINT@0,"NO. DE
    JOGADORES (1/2)";
270 INPUT TE$
280 IF TE$="2" THEN K=2
    ELSE K=1:X=70
```

Com estas linhas, o usuário pode escolher os tipos de jogo e implementar essa escolha tendo acesso ao estratagema de um jogador, entre as linhas 440 e 460, ou à versão padrão para dois jogadores, entre as linhas 500 e 570. Nossa estratégia está contida nestas linhas:

```
430 ON K GOTO 435,500
435 T$=INKEY$:T1=RND(2):
    IF T1=1 THENPT=1
    ELSE PT=-1
440 IF UM=PT THEN A=PT*J:
    B=PT*I: UM=0
450 IF PPOINT(X+A,Y+B)=3
    OR PPOINT(X+A,Y+B)=4
    THEN GOTO 540
455 T1=RND(2): IFT1=1
    THEN PT=1 ELSE PT=-1
460 A=A*PT: B=B*PT: D=A:
    A=B: B=D: IF PPOINT
    (X+A,Y+B)=3 OR PPOINT
    (X+A,Y+B)=4 THEN
    GOTO540
490 A=-A: B=-B: GOTO 540
```





Opção para um só jogador
Este fluxograma simplificado mostra a estrutura do programa com apenas um jogador. Cada processo é repetido no jogo completo para dois jogadores.

PLUS/4

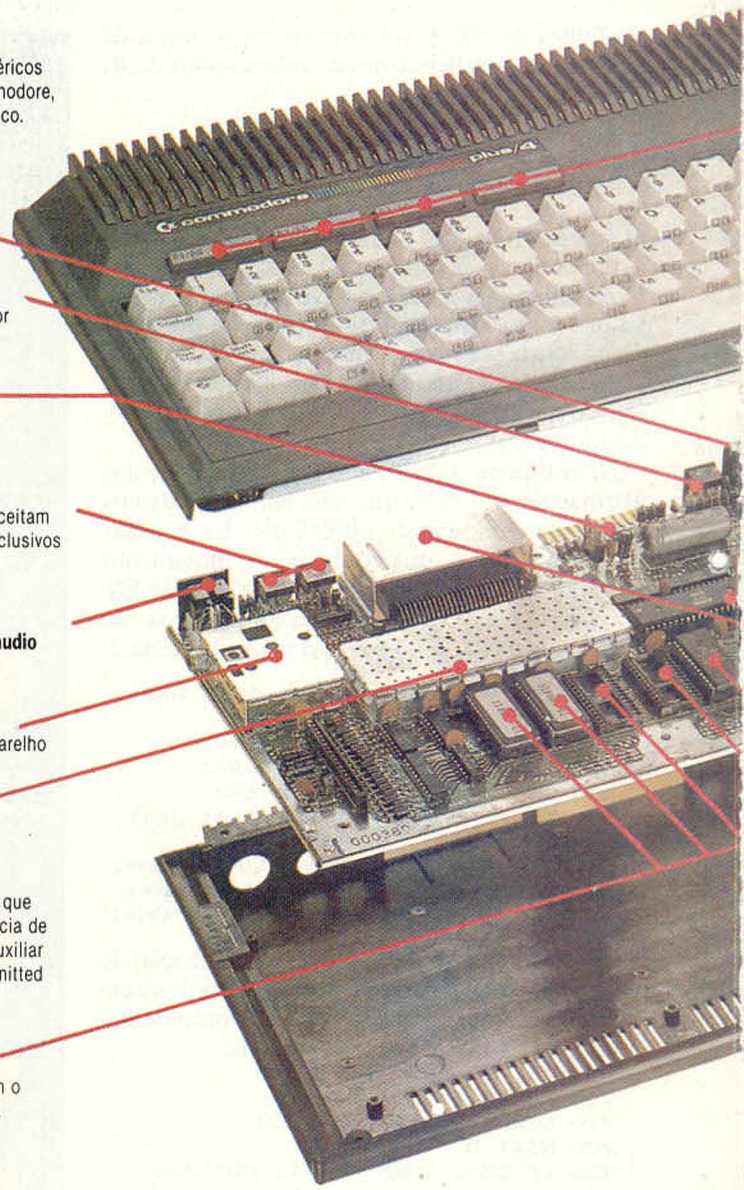
A Commodore submeteu o modelo 64 a aperfeiçoamentos e lançou o Plus/4, que dispõe de uma versão melhorada de BASIC, quatro programas aplicativos incorporados e 64 Kbytes de memória.

Modelo comercializado depois do Commodore 64, um dos micros mais vendidos no mundo, o Plus/4 utiliza o microprocessador 7501, desenvolvido a partir do 6502. Esse chip, acessando mais de 64 Kbytes de memória, dá espaço para um BASIC potente, enquanto mantém sua RAM livre para o usuário. Os comandos de gráficos e som são particularmente dignos de nota. No modo gráfico, o comando DRAW produz pontos ou linhas, e qualquer tipo de contorno pode ser preenchido com cores, por meio do comando PAINT.

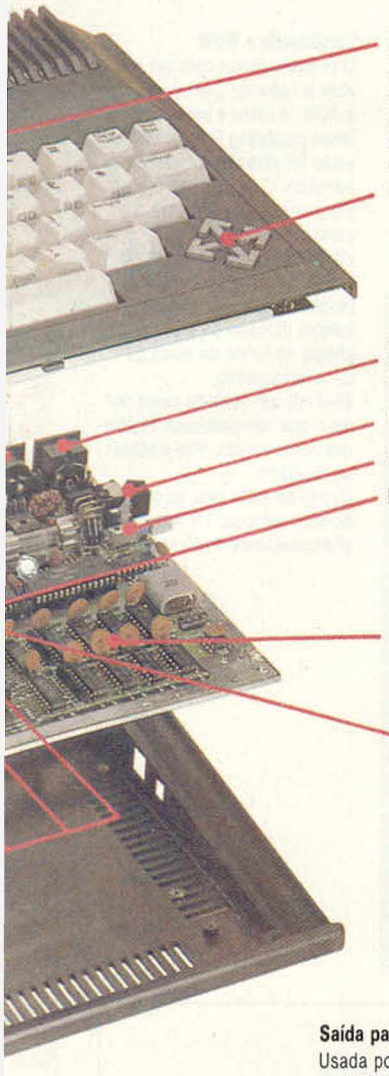
Outro comando, BOX, desenha quadrados e retângulos preenchidos com cores ou apenas seus contornos. Por fim, o versátil comando CIRCLE possibilita traçar círculos ou elipses, além de simples arcos, bastando em cada caso especificar a altura e o comprimento da curva.

Todos os comandos operam numa tela com resolução de 320 x 200 pixels, porém com opção para 121 cores, criadas a partir de quinze tons básicos. No entanto, o Plus/4 não permite a produção de sprites.

Os sons são emitidos pelo alto-falante do monitor de vídeo, em dois canais, respeitando os



O esperado sucessor do Commodore 64 possui todas as características que estão se tornando padrão na última geração de microcomputadores: o agrupamento das teclas do cursor, a memória de grande capacidade e os aplicativos incorporados. Claro sinal de que a Commodore não pretende perder sua posição de liderança neste competitivo mercado.



Teclas de função
Cada uma das quatro teclas pode ser programada com dois comandos de até 128 caracteres.

Teclas do cursor

Saída para fonte de alimentação

Chave liga/desliga

Botão de [Reset]

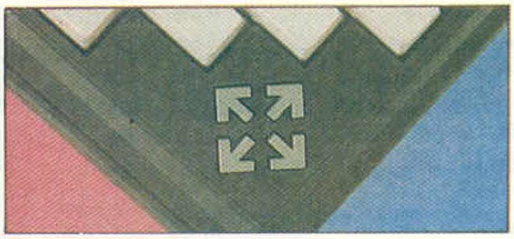
Microprocessador 7501

RAM de 64 K

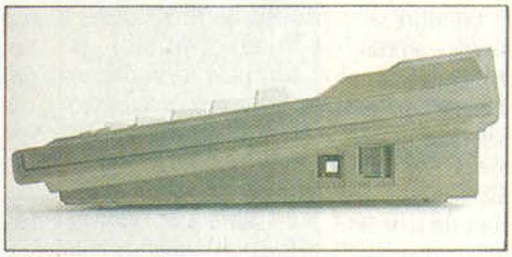
Saída para expansões
Usada por aplicativos em cartucho e por uma unidade de disco "rápida".

Teclas de função

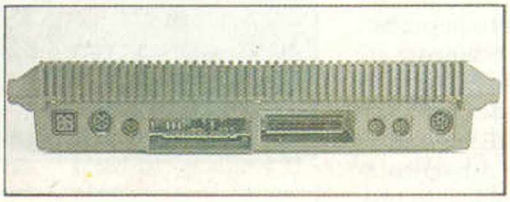
Teclas do cursor



Botão de [Reset]



Saída para expansões



Inovações
Ao contrário dos modelos antigos da Commodore, o Plus/4 possui as teclas [Escape] e [Reset]. Outras inovações são as teclas para movimentação do cursor agrupadas, as teclas de função (programáveis em BASIC) e a tecla [Help]. As saídas para fonte de alimentação, cassete, joysticks e expansões são todas diferentes das encontradas no Commodore 64 e Vic-20.

COMMODORE PLUS/4

MICROPROCESSADOR
7501.

CLOCK
0,9 ou 1,8 MHz.

MEMÓRIA
64 K de ROM, 64 K de RAM.

SISTEMA OPERACIONAL
DOS.

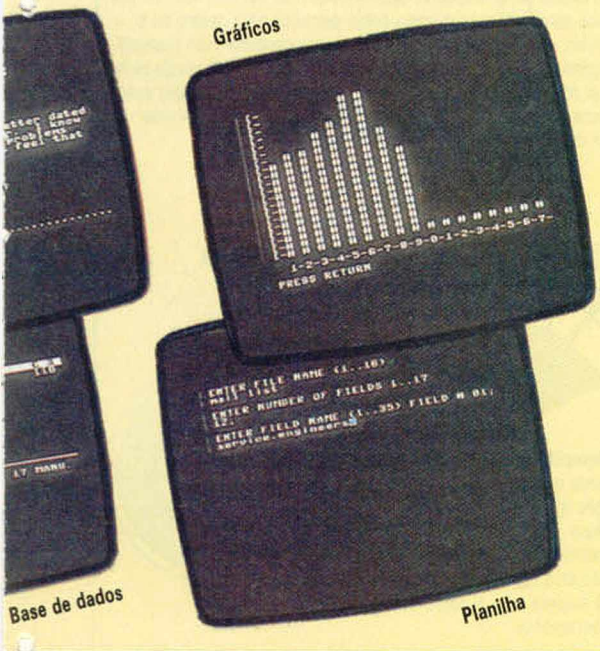
VÍDEO
Saída para monitor e TV comum.
Modo texto: 25 linhas de quarenta caracteres.
Modo gráfico: 320 x 200 pixels, em 121 cores.

TECLADO
Tipo QWERTY, com 67 teclas de função e quatro teclas de cursor, agrupadas em forma de diamante.

LINGUAGENS
BASIC.

PERIFÉRICOS
Interface serial e paralela. Saídas para joysticks, impressora matricial — em preto e branco ou em cores —, impressora margarida e plotter em quatro cores.

DOCUMENTAÇÃO
Manuais razoáveis que ensinam a programar e a usar os aplicativos.



Custo oculto
O software incorporado à ROM do Plus/4 inclui um processador de texto, planilha eletrônica, banco de dados e gerador de gráficos. Apesar da grande vantagem de estarem integrados, esses aplicativos só podem ser implementados com uma unidade de disco, o que aumenta bastante o custo da máquina.

controles que não fogem ao padrão normal. O maior aperfeiçoamento reside no corpo central da linguagem BASIC: comando AUTO, para produzir linhas numeradas automaticamente; RENUMBER, para renumerar as linhas do programa; e VERIFY, para conferir se o programa foi gravado no cassete ou no disco.

No modo texto, o Plus/4 mostra no vídeo 25 linhas de quarenta caracteres. O usuário pode especificar dois pontos na tela a fim de marcar as margens de uma "janela", dentro da qual vão aparecer todos os textos.

A quantidade de memória permite que o Plus/4 tenha quatro programas aplicativos incorporados no chip da ROM, mas que exigem a utilização de disquetes: processador de texto, planilha eletrônica, banco de dados e gerador de gráfico — planejados para execução integrada.

Muitos usuários ficarão mais entusiasmados com o Tedmon — o programa monitor em código de máquina — do que com os aplicativos, pois ele é de grande utilidade para se programar diretamente em linguagem de máquina.

A TROCA DO CHIP

Saber montar cabos e conectores é fundamental em eletrônica. Convém, portanto, aprender a dessoldar, mudar componentes e remover um chip de ROM, substituindo-o por um soquete para facilitar sua troca.

Depois de aprender a soldar, o passo seguinte será descobrir como desfazer as conexões soldadas, de modo bem-acabado e sem confusão. Quando estamos lidando só com cabos e conectores, não precisamos preocupar-nos demais — na certa será mais simples cortar o cabo, jogar fora o conector e substituí-lo.

Mas o que acontece quando se trata de trocar ou mudar de lugar um chip? Mesmo os transistores simples têm três terminais. Para remover um transistor de sua placa de circuito impresso, precisamos aquecer os três simultaneamente até o ponto de fusão da solda, de modo que o transistor possa ser liberado; ou temos de tirar a solda de cada um dos pinos por vez. E, se a idéia de desligar três pinos de uma vez já é desencorajadora, o que dizer dos quarenta pinos do processador típico de 8 bits?

Extração da ROM

Em vez de escolher um chip ao acaso para ser substituído (ou acrescentar chips de RAM para expandir a capacidade de uma máquina), vamos executar um projeto um pouco mais ambicioso: substituir a ROM do BASIC do ZX81 por um soquete — ligando um cabo multivias entre as conexões originais do chip e um pedaço de Protoboard, onde instalamos o soquete para receber a ROM que extraímos. A razão de escolhermos o ZX81 como objeto deste exercício está no fato de que o BASIC pode ser substituído pelo FORTH como linguagem residente (em ROM). Além da nova linguagem, a ROM do FORTH também incorpora um sistema operacional multitarefa, permitindo que mais de um programa rode ao mesmo tempo, cada qual inteiramente independente dos demais. Essa é uma conquista notável para uma máquina tão pequena, embora ela necessite de um mínimo de 2 Kbytes de RAM, o que pode exigir o acréscimo de um módulo de memória.

Além de ser um exercício das técnicas que até agora examinamos, um pequeno projeto como este possibilita o treino na manipulação de componentes e mostra a você como é necessário ser cuidadoso e preciso. Para testar o serviço pronto, recomenda-se o uso de um multímetro.

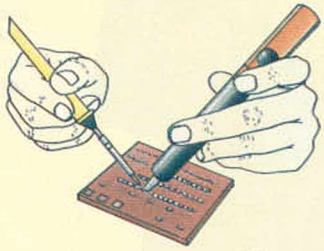
Receita

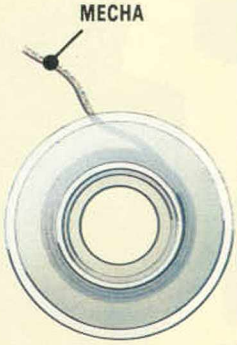
Se um chip tiver de ser substituído ou reposicionado, ou se for necessário colocar um novo chip, será indicada a instalação de um soquete. Os chips com soquete podem ser substituídos em pouco tempo. Há à venda sofisticados suportes para chips, chamados ZIF (Zero Insertion Force), que reduzem ao mínimo o risco de amassar um pino. Você precisará também de um carretel de mecha e de outras ferramentas, para a dessoldagem. O exercício que escolhemos é a retirada da ROM BASIC do ZX81 e sua substituição por um soquete padrão, do qual será puxado um cabo plano de fora da máquina para o soquete. O objetivo será substituir o BASIC do ZX81 pelo FORTH-in-ROM multitarefa, da David Husband — dando ao usuário, ao mesmo tempo, a opção de posteriormente retornar ao BASIC. Se você quiser trabalhar acompanhando este exemplo, precisará de pedaços de Protoboard, de cabo de 28 fios e de um pedaço de poliestireno.

Localizando a ROM
O primeiro passo consiste em abrir a caixa do ZX81 e localizar a ROM. A caixa é presa por cinco parafusos Philips; três estão localizados sob os pés de borracha. O pé que não contém um parafuso é o que está mais próximo aos soquetes EAR e MIC. Levante com cuidado os outros três e retire os parafusos. Levante a face inferior da caixa para chegar ao fundo da placa de circuito impresso.
Ela pode ser solta da caixa por meio dos três parafusos Philips que estão à vista. Vire a placa ao contrário.
O chip da ROM está localizado acima e um pouco à esquerda dos conectores do teclado.

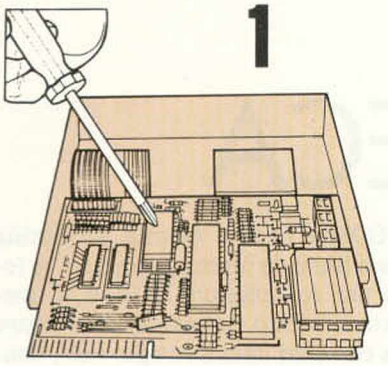
Ferramentas para dessoldagem

Há dois instrumentos patenteados para a dessoldagem: a "mecha para solda" — um fio de cobre muito fino, trançado numa fita e impregnado com solvente, cuja ação se apóia no fenômeno da função capilar (uma tensão de superfície que faz líquidos subirem por tubos estreitos) para puxar a solda derretida para dentro de si — e outro, considerado mais adequado, que consiste num dispositivo parecido com uma bomba de bicicleta, sugando em vez de expelir ar pela ponta. Esse instrumento opera com mais rapidez que a mecha, além de ter durabilidade bem maior. É essencial usar um dos dois para remover componentes.

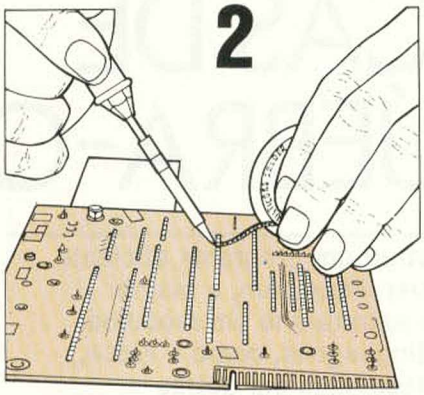




A dessoldagem
Aqueça com o soldador a junção até a solda derreter. Aplique a ferramenta de dessoldagem, pressione o botão acionador, e a solda será sugada para o corpo da ferramenta.

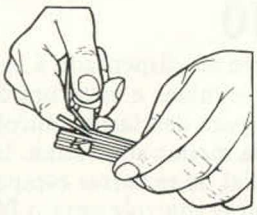


1



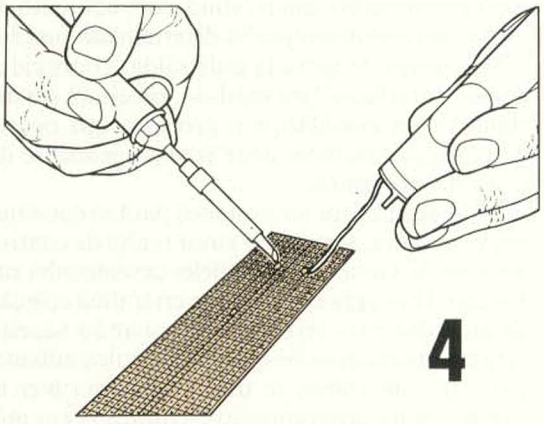
2

Retirando a ROM
Na dessoldagem, como na execução de junções, o segredo está na aplicação de calor suficiente. Com a ponta do soldador, coloque a mecha sobre a junção até que o líquido escorra. Você verá a mecha sugar a solda. Corte a extremidade da mecha de vez em quando, pois cada pequena parte fica logo saturada.



3

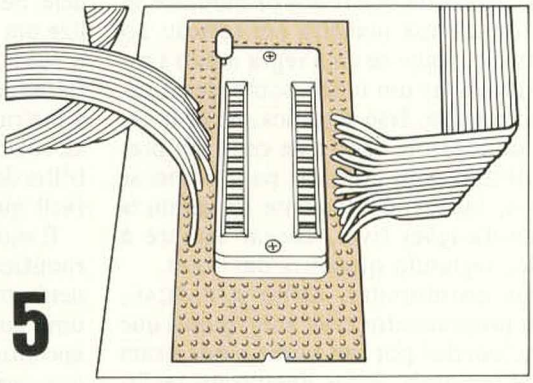
Cabo plano
Você vai usar o cabo multivias ou paralelo de 28 fios (ou dois com catorze fios). Descubra 1 cm da ponta de cada um dos 28 fios e estanhe cada um deles. Trabalhando a partir da ponta marcada com um pequeno recorte semicircular, solde o cabo plano no lugar, ao longo de ambos os lados. Todos os chips têm esses recortes, ou às vezes pontos para indicar o modo como devem ser alinhados.



4

Fazendo conexão
Corte os dois pedaços de cabo plano no tamanho correto, descasque e estanhe os 28 fios. Solde-os no Protoboard, assegurando-se de que os 28 fios da ponta marcada na área que mantinha o chip serão ligados na mesma ponta do soquete. Nenhum dos fios pode ficar cruzado.

Protoboard
Este produto é projetado para a montagem de protótipos: é como uma placa de circuito impresso, com muitas carreiras de furos, ligados por trilhas de cobre. Cortando essas trilhas, criam-se conexões entre os componentes colocados na placa. É ideal para experiências e para montagens que normalmente exigem muitas correções.

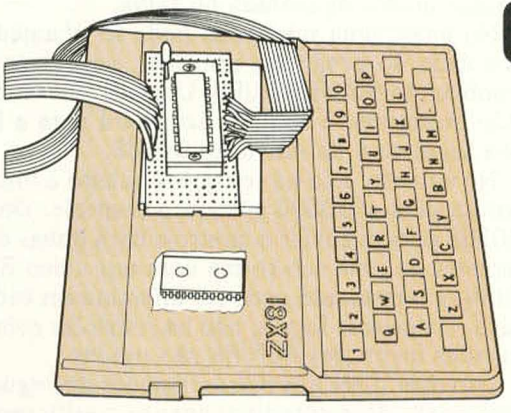


5

Montando o soquete
Para esta finalidade, o Protoboard é o meio mais apropriado. Corte um pedaço de quinze pontos de contato de largura por 25 cm de comprimento. Coloque o soquete do chip de modo que ele enquadre o isolamento central; solde um pino de canto para prendê-lo firmemente. Então trabalhe um lado de cada vez.

AVISO

A garantia de seu computador – se ainda estiver em vigor – poderá ser cancelada se pessoa não autorizada abrir a máquina.



6

Acabamento
Quando todas as conexões estiverem feitas, verifique cada uma delas, tanto na placa do computador como na nova, para assegurar-se de que estão separadas. Ao trabalhar um componente como este, com muitas conexões próximas, a possibilidade de curto-circuito é grande. Verifique as conexões com uma lente de aumento e, se estiver em dúvida, raspe entre os vãos com um estilete ou outro instrumento de metal.

PEÇAS DE QUEBRA-CABEÇA

Para construir um programa eficiente em qualquer linguagem, a melhor maneira é estruturá-lo em módulos. Algumas linguagens, como a PASCAL, facilitam isso, mas em outras é preciso usar artifícios.

Em programação, um módulo é um conjunto de instruções que desempenha determinada função.

Os pontos de entrada e de saída, conhecidos como "interfaces" do módulo, precisam ser definidos com exatidão, e o processo que ocorre entre essas interfaces deve ser independente do resto do programa.

Módulos podem ser juntados para se construir um programa, sem que o autor tenha de controlar o modo como cada um deles desempenha sua função. O programador pode criar uma coleção de módulos para serem usados quando necessitar; em outras ocasiões, passa módulos adiante, para que um colega os use. Mas para obter as vantagens da programação estruturada em módulos precisamos tomar nota do fluxo de controle e do fluxo de dados ao escrever os módulos.

Para ter a certeza de que todos os módulos se comportam da mesma maneira em relação ao fluxo de controle, segue-se uma regra muito simples: todos devem ter um único ponto de entrada e uma única saída. Isso significa, na prática, que o fluxo de controle dentro de cada um precisa ser projetado com cuidado para começar num ponto e, independentemente de quantos loops ou ramificações tiver, chegar sempre à mesma saída, seguindo qualquer das rotas.

Linguagens estruturadas, como a PASCAL, permitem ao programador criar sub-rotinas que podem ser acionadas por um nome e que usam suas próprias variáveis. Essas linguagens facilitam ao programador a entrada ou saída de uma rotina (chamada "procedimento") por meio dos pontos únicos de entrada ou saída.

No BASIC uma sub-rotina pode ser chamada por meio do programa principal, usando-se a combinação GOSUB-RETURN. Uma vez que tenha sido executada, o controle retornará para a linha seguinte à da instrução GOSUB.

No entanto, não há restrições quanto à linha para a qual o GOSUB enviará o controle. Dois GOSUB podem enviar o controle para linhas diferentes de uma sub-rotina com um único RETURN, e o resultado pode ser diferente em cada caso. Da mesma forma, não há restrições sobre quantas instruções RETURN são usadas.

Isso quer dizer que o programador de linguagem BASIC deve autodisciplinar-se, verificando se todos os GOSUB para a mesma sub-rotina apontam para a linha de mesmo número e se toda sub-rotina tem somente um RETURN. O melhor é adquirir o hábito de marcar a primeira linha de todas elas com uma instrução REM, dando a cada qual um título, e usar aquela linha como ponto de entrada. Faça do RETURN a última linha de sub-rotina.

A regra do GOTO

Um cuidado especial deve ser dispensado à instrução GOTO, que pode arruinar a estrutura de um programa. Só a use para desviar o controle para uma linha dentro da mesma sub-rotina. Isso evita o perigo potencial de se deixar escapar um RETURN ou de passar o controle para o RETURN errado. Há ocasiões em que é necessário não executar algumas linhas de uma rotina. Nesse caso, dê um GOTO para a linha onde está o RETURN e não haverá problemas.

Usar o GOTO dentro de um loop é ainda mais perigoso. Se o controle sai do loop, o BASIC não sabe disso e presume que o resto do programa é o corpo daquele loop. Quando estiver num loop, nunca dê um GOTO para uma linha fora dele. Se um loop precisa terminar mais cedo, utilize um contador ou uma variável de teste para o valor final e dê um GOTO para linha de teste (a linha com o NEXT ou o WHILE). Como com a instrução RETURN, coloque o NEXT ou o WHILE sozinhos numa linha, para facilitar. Seguir a trilha da estrutura de um programa é muito mais fácil quando se evitam os GOTO.

É muito comum o controle se extraviar pelas ramificações. Assim, procure não permitir que nenhuma decisão envie o controle para fora de uma sub-rotina, a menos que seja chamando apropriadamente outra sub-rotina. Lembre-se de que cada uma delas tem um único ponto de saída e, então, certifique-se de que é possível seguir o fluxo de controle através de cada ramificação até aquele ponto. Desenhar um fluxograma para a rotina torna mais fácil a checagem desse aspecto. Colocar uma sinalização (flag) pode reduzir a necessidade de GOTO em rotinas que envolvem loops e ramificações.

Para que os módulos possam ser utilizados como independentes uns dos outros, você deve projetá-los de maneira que a única influência de um sobre o outro seja por intermédio dos dados que trafegam entre eles. O programa principal passa os dados a um módulo e, quando este tiver sido executado, qualquer resultado pode ser passado de volta.

Os dados circulam pelos programas dentro de variáveis e a liberdade de movimento de uma va-

riável é chamada "raio de ação". Muitas linguagens de programação restrigem o raio de ação de uma variável a determinada sub-rotina. Em PASCAL as variáveis usadas em determinada sub-rotina (procedimento) devem ser declaradas para aquele procedimento. Variáveis declaradas para o programa principal são globais e são usadas em qualquer lugar do programa (inclusive dentro de qualquer de seus módulos). Variáveis declaradas num procedimento específico, contudo, são locais e só podem ser usadas dentro dele.

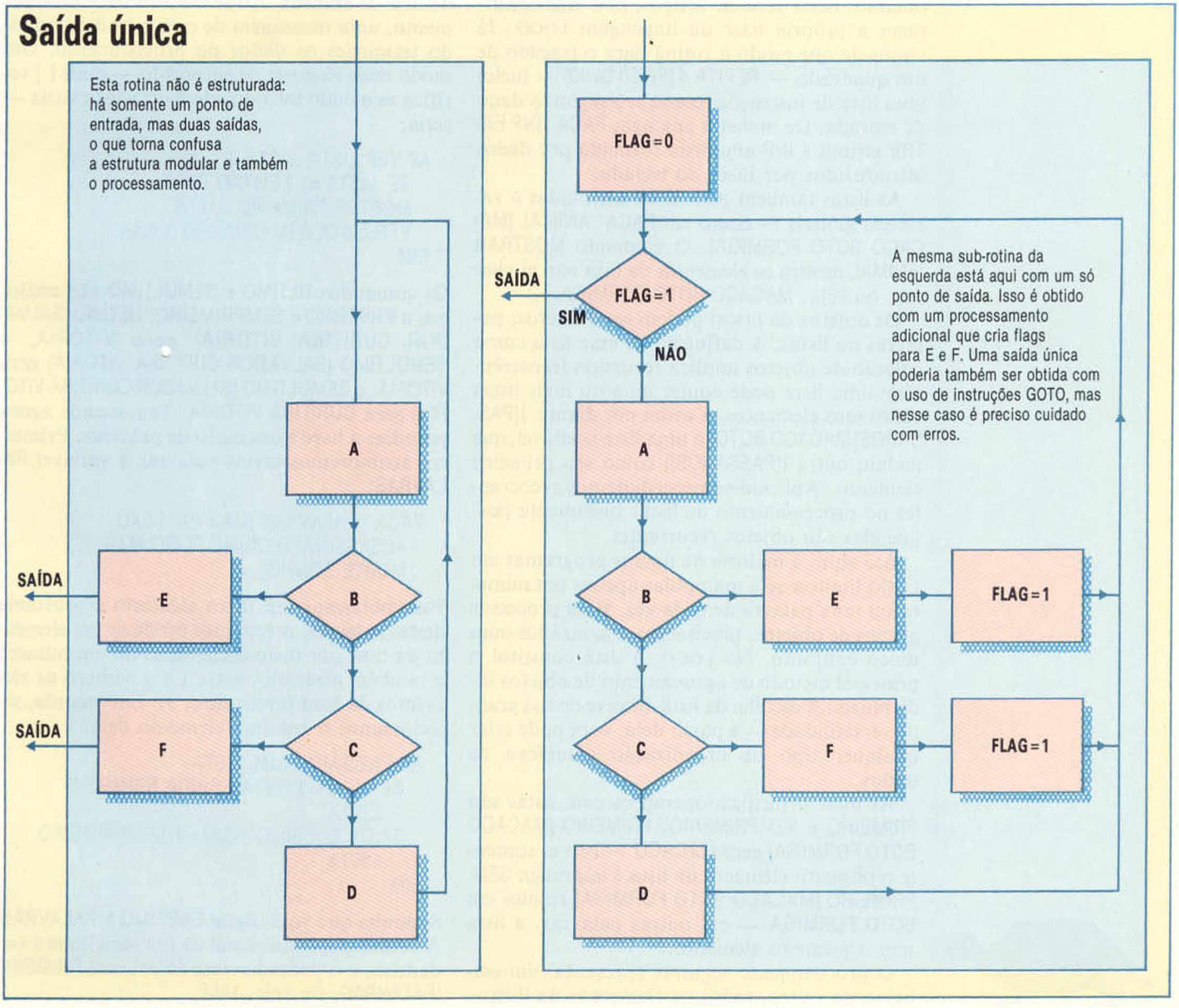
As variáveis locais, às vezes, têm o mesmo nome das globais, e usar uma delas não afeta o valor de outra. Uma linguagem que emprega variáveis locais permite escrever sub-rotinas sem preocupação quanto à influência que as variáveis de uma terão sobre as de outras.

Poucas versões da linguagem BASIC adotam variáveis locais, o que significa que, se quisermos escrever sub-rotinas independentes, devemos de alguma forma simular variáveis locais.

A maneira mais simples de conseguir isso é adotar nomes identificadores de variáveis que desempenham tarefas diferentes. Algumas convenções já existem e são usadas por programadores. Usar I, J e K como contadores de loops e valores de índices é muito comum.

Descrito o programa por um fluxograma, fica fácil numerar as sub-rotinas ou dar a elas outro tipo de identificação. Qualquer variável global que precise ser transformada em local em determinada sub-rotina ganha, então, como sufixo, o código correspondente à sub-rotina, para torná-la única. Assim, a rotina número 5 pode utilizar a variável local SOMA5 e TOTAL5, para distingui-las de SOMA12 e TOTAL12 na rotina número 12. Cuidado com a possibilidade de o BASIC que você está usando considerar somente os dois primeiros caracteres. As variáveis usadas para passar valores entre sub-rotinas e as usadas apenas no programa principal não precisam ser codificadas assim.

Saída única



POESIA ALEATÓRIA

Você imagina a tartaruga escrevendo poemas? Isso é o que conseguiremos que ela faça ao estudar as listas, um recurso essencial do LOGO. Antes, porém, vamos rever a recorrência e a livre associação de palavras.

Uma lista não passa de uma coleção ordenada de objetos. No LOGO, as listas sempre vêm escritas entre colchetes — por exemplo, [CURITIBA SALVADOR VITORIA]. Várias vezes nós as manipulamos nesta série de artigos, pois elas constituem a própria base da linguagem LOGO. Já vimos de que modo a rotina para o traçado de um quadrado — REPITA 4 [FR 50 DI 90] — inclui uma lista de instruções como seu segundo dado de entrada. De maneira análoga, FACA "INP ENTRE atribui a INP uma lista formada por dados introduzidos por meio do teclado.

As listas também podem ser atribuídas a variáveis globais — como em FACA "ANIMAL [MACACO BOTO FORMIGA]. O comando MOSTRAR :ANIMAL mostra os elementos da lista sem as chaves, ou seja, MACACO BOTO FORMIGA.

Os objetos do LOGO podem ser números, palavras ou listas. A definição de uma lista como coleção de objetos implica recursion (recorrência): uma lista pode conter uma ou mais listas como seus elementos, e assim por diante. [[PASSAROS] MACACO BOTO] é uma lista aceitável, que incluiu outra ([PASSAROS]) como seu primeiro elemento. Aplicam-se procedimentos recorrentes no processamento de listas justamente porque elas são objetos recorrentes.

Até aqui, a maioria de nossos programas em LOGO limitou-se a manipular apenas um número ou uma palavra de cada vez. Para processar grupos de objetos, precisamos organizá-los num único conjunto. No LOGO, a lista constitui o principal método de agrupamento de objetos individuais. A escolha da lista decorre de sua grande versatilidade — a partir dela, você pode criar qualquer tipo de organização complexa de dados.

As duas principais operações com listas são PRIMEIRO e SEMPRIMEIRO. PRIMEIRO [MACACO BOTO FORMIGA] gera MACACO — isto é, somente o primeiro elemento da lista é impresso. SEMPRIMEIRO [MACACO BOTO FORMIGA] resulta em BOTO FORMIGA — em outras palavras, a lista sem o primeiro elemento.

O procedimento seguinte apresenta, um embaixo do outro, todos os elementos da lista:

```
AP VER.LISTA :LISTA
  MOSTRE PRIMEIRO :LISTA
  VER.LISTA SEMPRIMEIRO :LISTA
FIM
```

Assim, VER.LISTA [SALVADOR CURITIBA VITORIA] imprime:

```
SALVADOR
CURITIBA
VITORIA
```

O comando inicial imprime o primeiro elemento da lista e depois transfere, para outra cópia de VER.LISTA, a tarefa de apresentação do resto da lista de entrada. Ao se executar esse procedimento, uma mensagem de erro é emitida quando terminam os dados do procedimento. Um modo mais elegante de encerrá-lo — onde [ ] verifica se o dado introduzido é uma lista vazia — seria:

```
AP VER.LISTA :LISTA
  SE :LISTA = [ ] ENTAO PARE
  MOSTRE PRIMEIRO :LISTA
  VER.LISTA SEMPRIMEIRO :LISTA
FIM
```

Os comandos ULTIMO e SEMULTIMO são análogos a PRIMEIRO e SEMPRIMEIRO. ULTIMO [SALVADOR CURITIBA VITORIA] gera VITORIA, e SEMULTIMO [SALVADOR CURITIBA VITORIA] gera VITORIA, e SEMULTIMO [SALVADOR CURITIBA VITORIA] gera CURITIBA VITORIA. Tentaremos agora parodiar a livre associação de palavras. Primeiro, atribuiremos certas palavras à variável PALAVRAS:

```
FACA "PALAVRAS [MAE PAI SEXO
  ASSASSINATO CIUME FOGO MAR
  MORTE SONHO]
```

Para obtermos um fluxo aleatório e constante destas palavras, precisamos produzir um elemento da lista por meio da geração de um número n também aleatório, entre 1 e o número de elementos da lista (neste caso, 9). Em seguida, selecionamos o enésimo elemento dela.

```
AP ENESIMO :NUM :LISTA
  SE :NUM = 1 ENTAO SAIDA PRIMEIRO
    :LISTA
  SAIDA ENESIMO :NUM - 1 SEMPRIMEIRO
    :LISTA
FIM
```

Suponha que você digite ENESIMO 1 :PALAVRAS. A declaração condicional na primeira linha é verdadeira, e o procedimento imprimirá PRIMEIRO :PALAVRAS, ou seja, MAE.

Experimente em seguida ENESIMO 2 :PALAVRAS — agora a declaração condicional é falsa, e o procedimento passa a ENESIMO 1 SEMPRIMEIRO :PALAVRAS. O primeiro elemento da lista é ignorado, e o procedimento escreve a primeira palavra dentre os elementos restantes — PAI. Assim, o procedimento para a geração de uma palavra aleatória a partir daquele vocabulário seria:

```
AP GERAR :LISTA
  SAIDA ENESIMO ((SORTEIE 9) + 1) :LISTA
FIM
```

Para rodá-lo, digite GERAR :PALAVRAS.

Esse procedimento restringe-se a uma lista de nove elementos. Se quisermos definir o número de itens que compõem determinada lista, usamos a seguinte rotina:

```
AP TAMANHO :LISTA
  SE :LISTA = [ ] ENTAO SAIDA 0
  SAIDA 1 + TAMANHO SEMPRIMEIRO :LISTA
FIM
```

Digite TAMANHO [FICCAO CIENTIFICA] para entender seu funcionamento. Como a lista contém várias palavras, a primeira declaração condicional é falsa, levando o procedimento a mostrar 1 + TAMANHO [FICCAO]. Agora o resultado de TAMANHO [FICCAO] é 1 + TAMANHO [ ]. Quando se chama TAMANHO e se introduz [ ], a declaração condicional da primeira linha torna-se verdadeira e o procedimento resulta em 0. Desta vez, TAMANHO [FICCAO] gera 0 + 1 = 1 e, por fim, TAMANHO [FICCAO CIENTIFICA] gera 1 + 1 = 2. Um procedimento mais geral para a geração de palavras aleatórias a partir de listas de qualquer tamanho pode ser:

```
AP GERAR :LISTA
  SAIDA ENESIMO ((SORTEIE TAMANHO
    :LISTA) + 1) :LISTA
FIM
```

Se você quiser uma seleção de dez comentários, basta digitar:

```
REPITA 10 [MOSTRE GERAR :PALAVRAS]
```

Esses programas para o processamento de listas possuem estrutura semelhante à de vários procedimentos para desenhos que vimos antes:

- Quando a tarcfa é muito simples, realize-a e pare.
- Caso contrário, complete apenas pequena parte da tarefa.
- A seguir, transfira o restante da tarefa para outro procedimento (em geral, uma cópia do procedimento original).

Esta estratégia é de grande eficácia, sendo freqüentemente utilizada nos programas de processamento de listas. Compare um programa para o traçado de um quadrado "cheio" (totalmente preenchido com linhas), como este:

```
AP QUADRADO :N
  SE :N = 0 ENTAO PARE
  REPITA 4 [FR :N DI 90]
  QUADRADO :N - 1
FIM
```

com a versão de LISTAR já apresentada. Observe a perfeita equivalência das estruturas dos dois procedimentos.

Quando não é possível a utilização da condicional na primeira instrução, o programa será executado indefinidamente, pois não existirá o comando primitivo PARE:

```
AP POLI :N
  FR 30 DI (360/ :N)
  POLI :N
FIM
```

O programa acima desenha um polígono com um número desejado de lados. Para parar a execução, utilize as teclas [CTRL] - G no MLOGO.

Poesia aleatória

Para escrever alguns poemas, precisamos gerar frases completas e não apenas palavras isoladas.

```
AP POEMA :TAMANHO
  SE :TAMANHO = 0 ENTAO MOSTRE ". PARE
  MOSTRAR"" '
  MOSTRAR GERAR :PALAVRAS
  POEMA :TAMANHO - 1
FIM
```

MOSTRAR"" ' serve para garantir o espaço entre as palavras. Para produzir uma frase de seis palavras com este procedimento, digite POEMA 6.

Um dos recursos para se ampliar a lista inicial de palavras, sem reescrevê-la, é a operação SENTENCA. Ela aceita a introdução de dois dados e os transforma numa lista. Dessa maneira, SENTENCA "GELEIA [POTE DE MEL] gera [GELEIA POTE DE MEL].

```
AP SOMA.PALAVRAS :LISTA
  FACA "PALAVRAS SENTENCA :LISTA
:PALAVRAS
FIM
```

Agora podemos ampliar PALAVRAS com SOMA.PALAVRAS [ANSIEDADE REPRESSAO [MEDO DE VOAR]]. Resta ainda a questão de saber se não se atribuiu, anteriormente, algum valor à variável PALAVRAS. O primitivo VALOR? verifica se isso ocorreu, respondendo VERD no caso de seu dado de entrada estar associado a algum valor:

```
AP SOMA.PALAVRAS :LISTA
  SE VALOR? :PALAVRAS ENTAO FACA
    "PALAVRAS [ ]
  FACA "PALAVRAS SENTENCA :LISTA
    :PALAVRAS
FIM
```

Empregando outra lista de palavras, criamos com esse procedimento o seguinte poema:

```
ESPECTRO RUIDOSAMENTE FALOU
ESPLENDIDO PARANOICO PLANETA
ATERRORIZADO O COM VERDE ESPECTRO
FLUTUANDO PARANOICO ROBO HOMEM VOOU
FALOU FLUTUANDO RUIDOSAMENTE
```

Uma falha óbvia deste poema é sua total desconsideração pela gramática. Talvez seja possível

Simplificação

ENTRE	
FACA	
PRIMEIRO	
SEMPRIMEIRO	SP
ULTIMO	
SEMULTIMO	SU
SENTENCA	SN
MOSTRE	MO

torná-lo mais compreensível se utilizarmos algumas estruturas sintáticas simples, como SUBSTANTIVO VERBO SUBSTANTIVO. Isso pode ser conseguido mediante o emprego de várias listas, uma para cada categoria gramatical. Seria possível, então, escolher uma palavra de cada lista de acordo com a estrutura gramatical desejada.

Ensinando sintaxe

Vamos ampliar as habilidades poéticas da tartaruga utilizando como exemplo uma estrutura gramatical do tipo VERBO SUBSTANTIVO. Começaremos definindo alguns conteúdos para VERBO e SUBSTANTIVO:

```
FACA "VERBO [COMPRAR ALUGAR VENDER]
FACA "SUBSTANTIVO [CASA MOTO SITIO
  CARRO]
```

Com estes conteúdos e o auxílio de dois novos programas, teremos um exemplo completo:

```
AP POEMA2 :ESTRUTURA
  SE :ESTRUTURA = [ ] ENTAO MOSTRE" '
  PARE
  POEMA2.1 PRIMEIRO :ESTRUTURA
  POEMA2 SEMPRIMEIRO :ESTRUTURA
FIM

AP POEMA2.1 :PL
  SE :PL = "SUBSTANTIVO ENTAO
    MOSTRAR" '
  MOSTRAR GERAR :SUBSTANTIVO
  SE :PL = "VERBO ENTAO MOSTRAR" '
  MOSTRAR GERAR :VERBO
FIM
```

Note que o programa GERAR, anteriormente definido, também foi utilizado. Para se rodar o programa POEMA2, devemos chamá-lo informando a estrutura gramatical desejada:

```
POEMA2 [VERBO SUBSTANTIVO]
```

A cada execução, você obterá uma frase diferente, como: ALUGAR SITIO, COMPRAR CASA etc. O programa POEMA2.1 poderia ser melhorado com o uso do comando primitivo VALOR. Para entender melhor este comando, siga o exemplo abaixo:

```
FACA "QUALQUER "OBJETO
FACA "OBJETO "CARRO
```

Com o auxílio do comando primitivo MOSTRE, podemos observar alguns resultados interessantes:

- MOSTRE :OBJETO resulta em CARRO;
- MOSTRE :QUALQUER resulta em OBJETO; e
- MOSTRE VALOR :QUALQUER resulta em CARRO.

O comando VALOR fez com que QUALQUER fosse substituído pelo seu valor (OBJETO); e, finalmente, é mostrado o valor de OBJETO, ou seja, CARRO. Utilizando esse conceito podemos reescrever o programa POEMA2.1:

```
AP POEMA2.1 :PL
  MOSTRAR " '
  MOSTRAR GERAR VALOR :PL
FIM
```

Como exercício, experimente criar poemas com uma estrutura gramatical mais complexa, como:

```
[ARTIGO SUBSTANTIVO ADJETIVO VERBO
  ADVERBIO PREPOSIÇAO ARTIGO
  SUBSTANTIVO ADJETIVO]
```

Para cada item faça uma lista específica com bastante criatividade. Você obterá poemas muito originais da tartaruga.

Respostas do exercício 8

A) Cálculo de potência:

```
AP POTENCIA :A :N
  SE NAO ((INT :N) = :N) ENTAO MOSTRE
  [SOMENTE NUMERO INTEIRO-EXPOENTE]
    PARE
  SE :N = 0 ENTAO SAIDA 1
  SAIDA :A * POTENCIA :A :N - 1
FIM
```

B) Converter para hexadecimal:

```
AP VER.HEXA :NUM
  SE :NUM < 10 ENTAO SAIDA :NUM
  SE :NUM = 10 ENTAO SAIDA "A
  SE :NUM = 11 ENTAO SAIDA "B
  SE :NUM = 12 ENTAO SAIDA "C
  SE :NUM = 13 ENTAO SAIDA "D
  SE :NUM = 14 ENTAO SAIDA "E
  SE :NUM = 15 ENTAO SAIDA "F
FIM

AP HEXA :NUM
  SE :NUM = 0 ENTAO PARE
  HEXA QUOC :NUM 16
  MOSTRAR VER.HEXA RESTO :NUM 16
FIM
```

C) Testar se um número é par:

```
AP PAR? :NUM
  SE ((RESTO :NUM 2) = 0) ENTAO SAIDA
    "VERD
  SAIDA "FALSO
FIM
```

D) Calcular uma área usando o método de Monte Carlo:

```
AP MC
  DESENHE SEMT FACA "IN 0
  MC1 1000 10 100
  MOSTRAR [A AREA E]
  MOSTRE :IN
FIM

AP MC1 :NUM :XNUM :YNUM
  SE :NUM = 0 ENTAO PARE
  PONTO.ALE :XNUM :YNUM
  SE DENTRO? ENTAO FACA "IN :IN + 1
  MC1 :NUM - 1 :XNUM :YNUM
FIM

AP PONTO.ALE :XNUM :YNUM
  DEFXY SORTEIE :XNUM :YNUM
FIM

AP DENTRO?
  SE CORY < CORX * CORX ENTAO SAIDA
    "VERD
  SAIDA "FALSO
FIM
```

PROGRAMAS COMPATÍVEIS

Ao tornar-se padrão industrial em sistemas operacionais, o Programa de Controle para Microprocessadores (CP/M) mudou a vida de seu criador, Gary Kildall, um dos fundadores da Digital Research.

Gary Kildall, funcionário da Intel americana, que desenvolveu o microcomputador 8080, criou a primeira versão de seu sistema CP/M (Control Program/Monitor) em 1974. Destinava-se a apoiar o compilador do PL/M, a primeira linguagem de alto nível produzida pela Intel.

No ano seguinte, Kildall acrescentou-lhe um editor (ED), um assembler (ASM) e um debugger (DDT: debugador, eliminador de erros), oferecendo o novo sistema operacional à própria Intel, que o recusou. Assim, em sociedade com Dorothy McEwan, começou a publicar revistas de informática e a vender diretamente seu Control Program/Monitor.

Intencionalmente ou não, Kildall havia criado um sistema que diminuía em muito o maior problema do microcomputador em seus primeiros anos de existência: compatibilidade. Os três computadores de maior popularidade no fim da década de 70 (PET, Apple e Tandy) possuíam sistemas operacionais incompatíveis entre si e os produtores independentes de software tinham de optar por um dos formatos.

Isso implicava reescrever por completo o software para seu aproveitamento numa máquina diferente daquela para a qual fora projetado. Mas o CP/M modificou a situação: sua larga aceitação entre os fabricantes acabou criando um "padrão" de fato. Os que haviam escolhido os processadores Intel 8080 ou Zilog Z80 para seus produtos especificaram o CP/M porque ele proporcionava uma maneira simples de manipular o acesso a tela, impressora, discos, teclado e assim por diante. E, como sua popularidade aumentasse, uma quantidade cada vez maior de software foi lançada no mercado.

A princípio, apenas alguns usuários selecionados receberam licença de usar o Programa de Controle para Microprocessadores. Em 1976, sobrecarregado de pedidos, Kildall demitiu-se do cargo de professor de Ciência da Computação numa academia naval em Monterey e fundou a Digital Research em Pacific Grove, Califórnia.

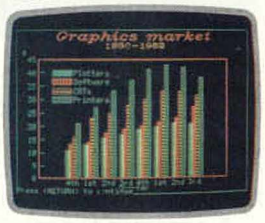
A empresa se ramifica

Enquanto crescia a aceitação do CP/M, a Digital Research voltou sua atenção para os sistemas de usuário múltiplo e criou o MP/M (Multi Program Monitor), que deveria ser compatível com o CP/M em todos os aspectos.

Suas primeiras versões, porém, não tiveram igual sucesso: o aumento da produção reduzira os custos dos microprocessadores para o consumidor final e a partilha de um deles entre diversos usuários deixou de fazer sentido econômico. Por esse motivo, mesmo revisto, o MP/M não se tornou popular.

Depois de levantar fundos junto a diversas empresas financeiras em 1981, a Digital Research tornou-se uma verdadeira multinacional, com forte presença na Europa, especialmente na Inglaterra, Alemanha e França.

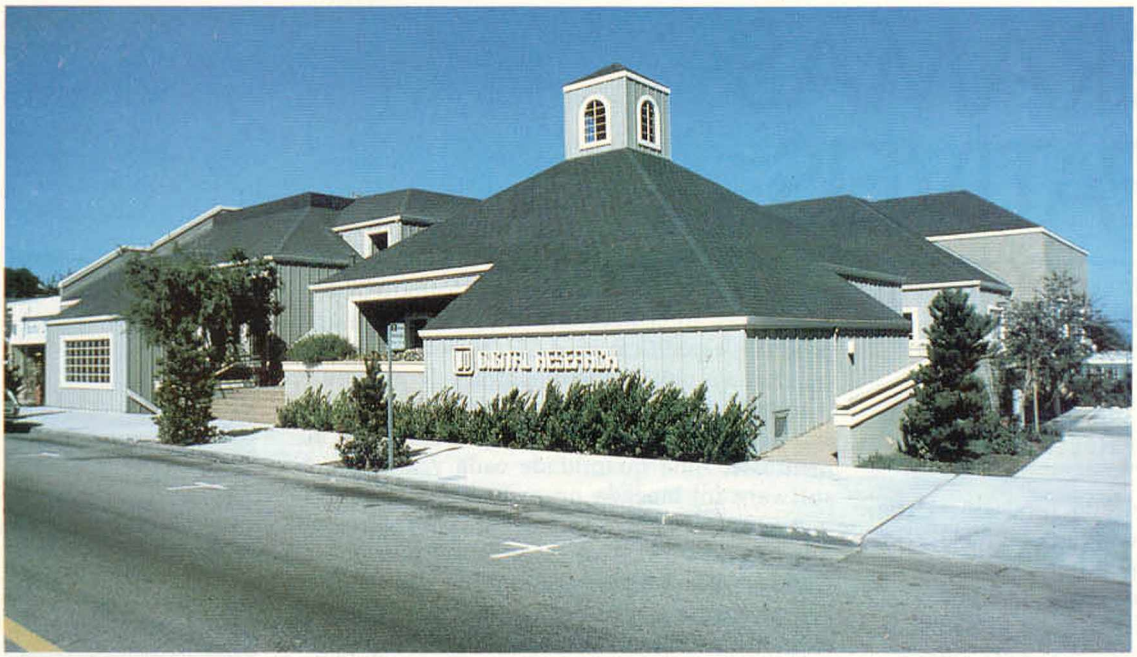
Projetos LOGO
A Digital Research entrou no campo das linguagens com seu DR LOGO. Como outros, este tem nos gráficos um de seus pontos fortes.



Gráficos comerciais
GSX é um pacote de software gráfico pioneiro, projetado para transferir gráficos entre as diferentes máquinas e aplicações.

John Rowley, presidente da Digital Research

Gary Kildall



Sede da Digital Research em Massachusetts, EUA

Ao mesmo tempo, constituía uma das empresas com maior possibilidade de ganhar o contrato para desenvolver um sistema operacional para o recém-criado microcomputador da International Business Machines, o IBM PC.

A encomenda acabou indo para a Microsoft; apesar disso, a Digital Research não parou de crescer. Logo atualizou o CP/M para os processadores Intel 8088/8086, de forma a fazê-los muito parecidos com o MS-DOS, e deu também um passo adiante com o CP/M Concurrent.

O CP/M Concurrent é o inverso do MP/M, permitindo que diversos programas diferentes sejam executados simultaneamente. Com esse programa, um usuário pode trabalhar com três coisas diferentes ao mesmo tempo — por exemplo, planilha eletrônica, geração de relatório e editor de texto —, passando de uma para outra à vontade.

As versões existentes do CP/M Concurrent podem exibir cada uma das telas — ou apenas parte delas — simultaneamente, usando "janelas"; os novos modelos em desenvolvimento prometem executar diretamente a maioria dos programas escritos para o IBM PC-DOS.

O "microware"

Entre as decisões estratégicas que a Digital Research e muitas outras empresas de sistemas tomaram está a de passar todo o seu trabalho de desenvolvimento para a linguagem C, particularmente notável pela portabilidade. O código escrito em C precisa apenas ser recompilado para se tornar útil em outro processador, embora seus adversários argumentem que é melhor fazer um trabalho apropriado em ASSEMBLER para cada processador isoladamente. Contudo, sua popularidade vem crescendo, e uma vez que o sistema operacional UNIX, amplamente utilizado, é escrito em C, a tendência a usar esta linguagem parece ser irreversível.

A Digital Research tem sido coerente em seu ponto de vista de que a verdadeira portabilidade só é possível por meio das linguagens de alto nível, das quais fornece grande variedade. Na faixa de micros, desenvolveu um Personal BASIC, um Personal CP/M e sua própria versão de LOGO. O Personal CP/M, como o CP/M-86, é projetado para ser armazenado em ROM. A Digital descreve isso como sendo "microware".

VIP e GSX

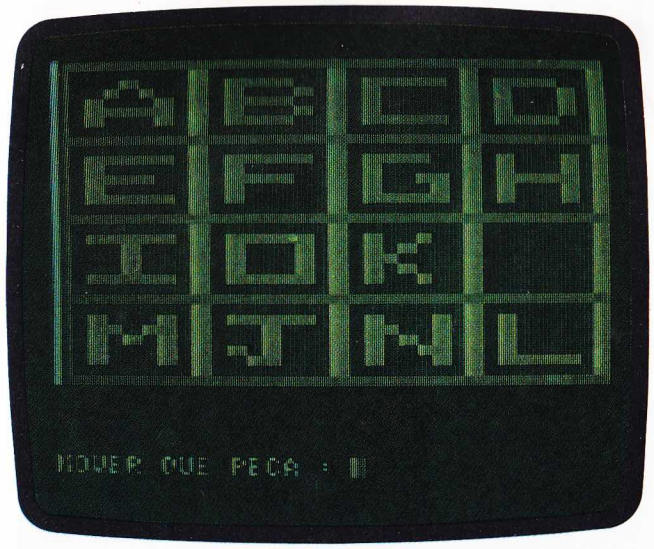
Em fins de 1984, a empresa de Gary Kildall dedicava-se aos projetos do VIP e do GSX. O VIP é um gerador de telas padronizadas que permite a apresentação de uma interface uniforme ao usuário, independentemente do pacote de aplicações que estiver sendo executado.

Várias aplicações podem usar as mesmas informações. Nesse aspecto, o VIP é semelhante à tecnologia do Lisa e do Macintosh da Apple, só que exige bem menos memória. O VIP pode ser executado em qualquer computador com mais de 50 Kbytes de RAM e equipado com espaço de disco de 150 Kbytes ou mais.

O GSX faz pelos gráficos o que o CP/M faz pelos discos. Usa um conjunto padrão de funções gráficas, destinadas a várias peças diferentes de hardware.

Um programa GSX pode ser executado numa tela em cores ou em preto e branco, numa impressora matricial e num plotter, sem qualquer modificação. Além da relativa falta de software, havia também dificuldades na criação de novos padrões de gráficos, pois o sistema não produz a mesma qualidade dos programas escritos para uma única máquina.

Depois de lançar produtos como GSX, VIP e LOGO, e com potencial para concorrer com a Microsoft no campo de softwares aplicativos, a Digital Research coloca-se como uma das maiores firmas internacionais do ramo.



Visão de conjunto
Em algumas situações, o jogador pode sentir-se tentado a preencher, de imediato, a célula desocupada com a peça correta. Esse procedimento, no entanto, nem sempre convém, pois acaba exigindo maior número de movimentos. Antes de mover a peça, verifique o conjunto, para saber os movimentos seguintes.

Arranjos distintos
A cada nova tentativa, o programa dispõe as peças de forma diferente, criando arranjos mais ou menos difíceis. Assim, o resultado de uma tentativa raramente será igual ao de outra.

```
2000 REM *** COLOCAR VARIAVEIS ***
2005 DEF FN C(X) = (X-1) * 8 + 2
2010 DIM B(4,4): FOR I = 1 TO 4: FOR J =
     1 TO 4: K = K + 1: B(J,I) = K: NEXT
     J,I
2020 DIM DIR(4,2): FOR I = 1 TO 4: READ
     DIR(I,1),DIR(I,2): NEXT: DATA
     1,0,0,1,-1,0,0,-1
2025 GR: COLOR = 12
2026 FOR I = 1 TO 32: HLIN 0,32 AT I:
     NEXT
2027 COLOR = 15
2028 FOR I = 0 TO 32 STEP 8: HLIN 0,32
     AT I: VLIN 0,32 AT I:NEXT I
2029 CO = 15: GOSUB 2100
2030 VTAB 23: PRINT "<<< ESTOU
     MISTURANDO AS LETRAS >>>     "
2040 SX = 4: SY = 4: SC = INT ( RND (1)
     * 50) + 100: FOR K = 1 TO SC
2050 D = INT ( RND (1) * 4) + 1: PX = SX
     + DI(D,1): PY = SY + DI(D,2): IF
     PX < 1 OR PX > 4 OR PY < 1 OR PY >
     4 THEN 2050
2060 B(SX,SY) = B(PX,PY)
2061 J = SX: I = SY: CO = 14: GOSUB 2200
2062 J = PX: I = PY: CO = 12: GOSUB 2200
2065 B(PX,PY) = 0: SX = PX: SY = PY
2070 NEXT K: RETURN
2100 REM *** DESENHAR TABLADO ***
2110 FOR I = 1 TO 4: FOR J = 1 TO 4: CO
     = 14: GOSUB 2200: NEXT J,I: RETURN
2200 REM *** DESENHAR LETRA ***
2205 X = FN C(J): Y = FN C(I): COLOR =
     CO
```



```
2210 ON B(J,I) ; 1 GOSUB 2220,2221,2222,
     2223,2224,2225,2226,2227,2228,2229,
     2230,2231,2232,2233,2234,2235:
     RETURN
2220 RETURN
2221 PLOT X + 2,Y: PLOT X + 1, Y + 1:
     PLOT X + 3,Y + 1: VLIN Y + 2,Y + 4
     AT X: VLIN Y + 2, Y + 4 AT X + 4:
     HLIN X,X + 4 AT Y + 3: RETURN
2222 VLIN Y,Y + 4 AT X: HLIN X,X + 3 AT
     Y: HLIN X,X + 3 AT Y + 2: HLIN X,
     X + 3 AT Y + 4: PLOT X + 4,Y + 1:
     PLOT X + 4,Y + 3: RETURN
2223 HLIN X,X + 4 AT Y: HLIN X,X + 4 AT
     Y + 4: VLIN Y,Y + 4 AT X: RETURN
2224 HLIN X,X + 3 AT Y: HLIN X,X + 3 AT
     Y + 4: VLIN Y , Y + 4 AT X: VLIN
     Y + 1,Y + 3 AT X + 4: RETURN
2225 HLIN X,X + 4 AT Y: HLIN X,X + 3 AT
     Y + 2: HLIN X,X + 4 AT Y + 4: VLIN
     Y,Y + 4 AT X: RETURN
2226 HLIN X,X + 4 AT Y: HLIN X,X + 3 AT
     Y + 2: VLIN Y,Y + 4 AT X: RETURN
2227 HLIN X,X + 3 AT Y: HLIN X,X + 4 AT
     Y + 4: VLIN Y,Y + 4 AT X: VLIN Y +
     2,Y + 4 AT X + 4: PLOT X + 3,Y + 2:
     RETURN
2228 HLIN X,X + 4 AT Y + 2: VLIN Y,Y + 4
     AT X: VLIN Y,Y + 4 AT X + 4: RETURN
2229 HLIN X,X + 4 AT Y: HLIN X, X + 4 AT
     Y + 4:: VLIN Y,Y + 4 AT X + 2:
     RETURN
2230 HLIN X,X + 4 AT Y: HLIN X + 1,X + 2
     AT Y + 4: PLOT X,Y + 3: VLIN Y,Y +
     4 AT X + 2: RETURN
2231 VLIN Y,Y + 4 AT X: PLOT X + 1,Y + 2
     : PLOT X + 2,Y + 1: PLOT X + 2,Y +
     3: PLOT X + 3,Y: PLOT X + 3,Y + 4:
     RETURN
2232 HLIN X,X + 4 AT Y + 4: VLIN Y,Y + 4
     AT X: RETURN
2233 VLIN Y,Y + 4 AT X: VLIN Y,Y + 4 AT
     X + 4: PLOT X + 1,Y + 1: PLOT X +
     3,Y + 1: PLOT X + 2,Y + 2: RETURN
2234 VLIN Y,Y + 4 AT X: VLIN Y,Y + 4 AT
     X + 4: PLOT X + 1,Y + 1: PLOT X +
     2,Y + 2: PLOT X + 3,Y + 3: RETURN
2235 VLIN Y,Y + 4 AT X: VLIN Y,Y + 4 AT
     X + 4: HLIN X,X + 4 AT Y: HLIN X,X
     + 4 AT Y + 4: RETURN
2990 RETURN
3000 REM *** JOGAR ***
3005 NORMAL
3010 HOME: PRINT "MOVER QUE PECA: ";:GET
     ANS$: PRINT ANS$
3015 IF ANS$ < "A" OR ANS$ > "Z" THEN
     PRINT CHR$ (7): GOTO 3010
3020 FOR K = 1 TO 4: PX = SX + DIR(K,1):
     PY = SY + DIR(K,2)
3025 IF PX < 1 OR PX > 4 OR PY < 1 OR
     PY > 4 THEN 3040
3030 IF B(PX,PY) = ASC (ANS$) - 64 THEN
     MO = MO + 1: GOTO 3100
3040 NEXT K: PRINT CHR$ (7): GOTO 3010
3100 B(SX,SY) = B(PX,PY): J = SX:I = SY
     CO = 14: GOSUB 2200
3110 J = PX: I = PY: CO = 12: GOSUB 2200
     : B(PX,PY) = 0: SX = PX: SY = PY
3200 K = 0: FOR I = 1 TO 4: FOR J = 1 TO
     4: K = K + 1: IF K = 16 THEN K = 0
3210 IF B(J,I) = K THEN NEXT J,I: RETURN
3220 GOTO 3010
3990 RETURN
4000 REM *** FIM ***
4010 TEXT: HOME
4020 PRINT "VOCE CONSEGUIU !!! "
4030 PRINT "AS LETRAS FORAM TROCADAS "
     SC" VEZES."
4035 PRINT "VOCE RESOLVEU EM "MO"
     MODIFICACOES."
4037 PRINT : PRINT "QUER TENTAR DE NOVO
     ? (S/N)";:GET Q$
4040 IF Q$ = "S" THEN RUN
4050 RETURN
```

BLOQUEIE
PARA LINHA APPLE

Este jogo para duas pessoas pode se tornar bastante disputado se os adversários forem dotados de boa coordenação. Após a exibição das instruções na tela, uma música marca o início do jogo. Aparecem então números em tamanho grande, indicando uma contagem regressiva de 5 até 1. Este último algarismo permanece na tela, fazendo as vezes de obstáculo. Cada jogador dispõe de quatro teclas — W, A, S, Z e I, J, K, M — para controlar as linhas que vão sendo traçadas pelo programa. São duas linhas de cores diferentes, uma para cada jogador, e, à medida que o tempo passa, são desenhadas mais rapidamente. Perde o jogo quem tocar o obstáculo ou a linha adversária.

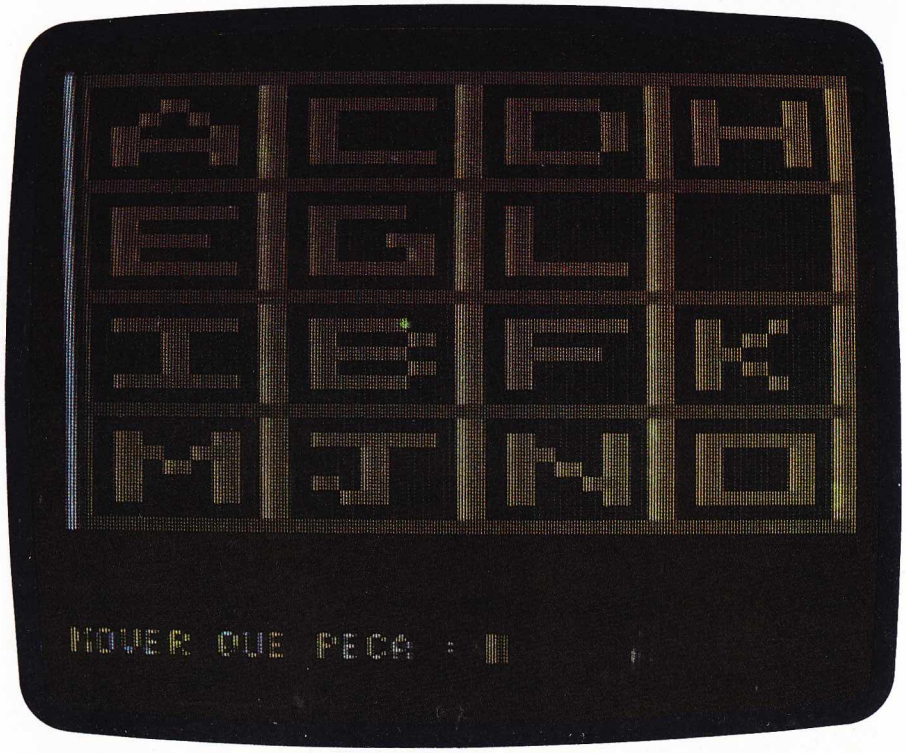
```
5    REM  *** PARA A LINHA APPLE ***
10   REM  ********************
11   REM  ***                 ***
12   REM  ***  BLOQUEIE       ***
13   REM  ***                 ***
14   REM  ********************
15   REM
16   REM
20   GOSUB 1000: REM INSTRUCOES
30   GOSUB 2000: REM INICIO
40   GOSUB 3000: REM JOGO
50   GOSUB 4000: REM FIM
60   END
1000 REM ******************
1001 REM *** INSTRUCOES ***
1002 REM ******************
1010 TEXT : NORMAL : HOME
1020 VTAB 2: HTAB 12: PRINT "***
     BLOQUEIE ***"
1030 VTAB 5
1031 PRINT "NESTE JOGO, DOIS JOGADORES
     CONTROLAM A CRIACAO DE UMA LINHA."
```

```
1032 PRINT
1040 PRINT "O PRIMEIRO JOGADOR CUJA
     LINHA BATER NO MURO OU NA LINHA
     DO ADVERSARIO PERDE O JOGO. "
1050 PRINT: PRINT "JOGADOR #1
      DIRECAO      JOGADOR #2": PRINT
1051 PRINT "    W                  ACIMA
                I": PRINT
1052 PRINT "A      S       ESQUERDA
      DIREITA  J     K": PRINT
1053 PRINT "     Z            ABAIXO
      N           M"
1090 VTAB 23: INPUT "TECLE RETURN PARA
     CONTINUAR : ";RES$
1990 RETURN
2000 REM **************
2001 REM *** INICIO ***
2002 REM *** MUSICA INICIAL ***
2010 POKE 768,173: POKE 769,48: POKE
     770,192: POKE 771,136: POKE 772,
     208: POKE 773,4: POKE 774,198:
     POKE 775,1: POKE776,240
2015 POKE 777,8: POKE 778,202: POKE
     779,208: POKE 780,246: POKE 781,
     166: POKE 782,0: POKE 783,76:POKE
     784,0: POKE 785,3: POKE 786,96
2016 REM *** DELIMITA A TELA ***
2020 GR : HOME : COLOR = 15: HLIN 0,39
     AT 0: HLIN 0,39 AT 39: VLIN
     0,39 AT 0: VLIN 0,39 AT 39
2021 PRINT "JOGADOR #1
              JOGADOR #2"
2025 X1 = INT ( RND (1) * 5) + 1:
     Y1 = INT ( RND (1) * 10) + 10:
     X2 = 38 - INT ( RND (1) * 5):
     Y2 = INT ( RND (1) * 10) +10:
     D1 = 1: D2 = 3
2030 FOR I = 1 TO 6: READ N,D: POKE
     0,N: POKE 1,D: CALL 768: NEXT I:
     DATA 110,75,70,75,55,75,45,200,
     55,100,45,255
2040 REM *** CONTAGEM REGRESSIVA ***
2100 FOR I = 5 TO 1 STEP  -1: COLOR=0:
     FOR J = 18 TO 22: VLIN 18,24 AT J
     : NEXT : COLOR = 15: POKE 0,200:
     POKE 1,5: CALL 768
2110 ON I GOTO 2121,2122,2123,2124,2125
2121 HLIN 18,22 AT 24: HLIN 18,24 AT 20
     : HLIN 18,20 AT 18: GOTO 2130: REM
     *** NUMERO UM ***
2122 HLIN 18,22 AT 18: HLIN 18,22 AT 21
     : HLIN 18,22 AT  24: VLIN 18,21 AT
     22: VLIN 21,24 AT 18: GOTO 2130: REM
     *** NUMERO DOIS ***
2123 HLIN 18,22 AT 18: HLIN 18,22 AT 21
     : HLIN 18,22 AT 24: VLIN 18,24 AT
     22: GOTO 2130: REM *** NUMERO TRES
     ***
2124 HLIN 18,22 AT 21: VLIN 18,21 AT 18
     VLIN 18,24 AT 22: GOTO 2130 : REM
     *** NUMERO QUATRO ***
2125 HLIN 18,22 AT 18: HLIN 18,22 AT 21
     : HLIN 18,22 AT 24: VLIN 18,21 AT
     18: VLIN 21,24 AT 22: GOTO 2130: REM
     *** NUMERO CINCO ***
2128 REM *** PAUSA DA CONTAGEM REGRESSIVA ***
2130 FOR PA = 1 TO 300: NEXT PA,I
2132 REM *** APAGA ULTIMO NUMERO NA TELA ***
2135 COLOR = 0: FOR J = 18 TO 22: VLIN
     18,24 AT J: NEXT : COLOR = 15:
     POKE 0,200: POKE 1,5: CALL 768
2990 RETURN
3000 REM ************
3001 REM *** JOGO ***
3002 REM ************
3005 REM *** IMPRIME A LINHA NA TELA ***
3010 COLOR = 7 : PLOT X1,Y1: COLOR =9:
     PLOT X2,Y2
3015 NT = 5 +((MO < 35) * (30 - INT (MO
     / 15) * 15))
3018 REM *** PEGA O CODIGO ASCII DA TECLA
     PRESSIONADA ***
3020 FOR I = 1 TO NT:KEY = PEEK (-16384)
     : IF KEY < 128 THEN 3050
3025 POKE  - 16368,0
3028 REM *** DETERMINA AS TECLAS E SEUS
     MOVIMENTOS ***
3030 IF KEY = 193 THEN D1 = 3
3031 IF KEY = 215 THEN D1 = 4
3032 IF KEY = 211 THEN D1 = 1
3033 IF KEY = 218 THEN D1 = 2
3040 IF KEY = 202 THEN D2 = 3
3041 IF KEY = 201 THEN D2 = 4
3042 IF KEY = 203 THEN D2 = 1
3043 IF KEY = 205 THEN D2 = 2
3050 NEXT I
3090 REM *** ALTERA AS COORDENADAS DO
     JOGADOR 1 ***
3100 ON D1 GOTO 3110,3120,3130,3140
3110 X1 = X1 + 1: GOTO 3150
3120 Y1 = Y1 + 1: GOTO 3150
3130 X1 = X1 - 1: GOTO 3150
3140 Y1 = Y1 - 1: GOTO 3150
3145 REM *** VERIFICA SE O JOGADOR 1
     PERDEU A PARTIDA ***
3150 IF X1 < 1 OR X1 > 38 OR Y1 < 1 OR
     Y1 > 38 OR SCRN( X1,Y1) < > 0 THEN
     WL = 2: RETURN
3190 REM *** ALTERA AS COORDENADAS DO
     JOGADOR 2 ***
3200 ON D2 GOTO 3210,3220,3230,3240
3210 X2 = X2 + 1: GOTO 3250
3220 Y2 = Y2 + 1: GOTO 3250
3230 X2 = X2 - 1: GOTO 3250
3240 Y2 = Y2 - 1: GOTO 3250
3245 REM *** VERIFICA SE O JOGADOR 2
     PERDEU A PARTIDA ***
3250 IF X2 < 1 OR X2 > 38 OR Y2 < 1 OR
     Y2 > 38 OR SCRN ( X2,Y2 ) <> 0
     THEN WL = 1: RETURN
3300 POKE 0,90: POKE 1,30: CALL 768:
     MO = MO + 1: GOTO 3010
4000 REM ************
4001 REM *** FIM ***
4002 REM ************
4010 HOME : PRINT "A PARTIDA TERMINOU ."
     : PRINT " O JOGADOR NUMERO "WL" E' O
     VENCEDOR!!!"; CHR$ (7); CHR$ (7)
4990 RETURN
```


LETRAS MÁGICAS

PARA LINHA APPLE

À semelhança dos demais, Letras Mágicas é um quebra-cabeça que exige certa dose de paciência, raciocínio e atenção. O jogo consiste num tabuleiro de 4 linhas por 4 colunas (dezesseis células) e quinze peças compostas por letras de A a O. O programa desafia o jogador a colocar as peças em ordem alfabética, linha por linha. Para tanto, os movimentos permitidos são: para cima, para baixo e para os lados, sempre de uma célula para a vizinha que não estiver ocupada. Tente descobrir a melhor técnica de movimentação, a fim de ordenar as letras com o menor número possível de movimentos.




```
1     REM ***************************
5     REM *** PARA A LINHA APPLE ***
10    REM ***   LETRAS MAGICAS    ***
15    REM ***************************
20    GOSUB 1000: REM INSTRUCOES
30    GOSUB 2000: REM COLOCAR VARIAVEIS
40    GOSUB 3000: REM JOGAR
50    GOSUB 4000: REM FIM
60    END
1000 REM *** INSTRUCOES ***
1010 TEXT: HOME: NORMAL
1020 VTAB 3: HTAB 9: PRINT "***
     QUADRADOS MAGICOS ***"
1030 VTAB 7: PRINT "NESSE JOGO, VOCE TEM
     UM QUADRADO 4 POR 4 QUE CONTEM AS
     LETRAS DE A-O."
1040 PRINT: PRINT "O OBJETIVO E DE
     ALFABETIZAR HORIZONTALMENTE AS
     LETRAS QUE NAO ESTAO EM ORDEM."
1050 PRINT: PRINT "VOCE PODERA MEXER UMA
     DAS PECAS PARA OS LADOS OU PARA
     CIMA OU PARA BAIXO; MAS SO SE
     HOUVER UM ESPACO VAGO."
1060 VTAB 23: INVERSE: PRINT " APERTE
     QUALQUER TECLA PARA CONTINUAR:"
1070 IF PEEK ( -16384) > 128 THEN POKE
     - 16368,0: RETURN
1080 GOTO 1070
```